UVM | EDUCACIÓN CONTINUA

**Bootcamp:**

# App Developer

Programa *online* de 224 horas
(27 semanas)

# ¿Qué es un Bootcamp?

Una experiencia *learning by doing* significativa de aprendizaje, en la cual los participantes adquieren conocimientos y desarrollan competencias tecnológicas de aplicación inmediata, de una manera práctica y efectiva a través de sesiones sincrónicas semanales interactivas en línea y actividades de aprendizaje en la plataforma, guiados por profesores expertos y mediante modelo educativo integral.

# Características del programa:

**Modalidad Bootcamp interactivo de 224 horas:**

- 104 horas de clases de clases sincrónicas interactivas en línea (clases en vivo).
- 120 horas actividades asincrónicas.

**Las 104 horas de clases sincrónicas interactivas en línea (clases en vivo) se imparten en un período de 27 semanas:**

- Fase de Preparación
  - Semana 1: 2 horas de clases
  - Semana 2: 2 horas de clases
- Fase de Bootcamp
  - Semanas de la 3 a la 27 (total 25 semanas): 4 horas de clase por semana

**Las 120 horas de actividades asincrónicas consisten en:**

- Clases pregrabadas (una por semana)
- Ejercicios y casos prácticos (uno por semana)
- Foros (uno por semana)
- Exámenes de certificación (al final del Bootcamp)

# Objetivo:

El participante aprenderá cómo 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.

El participante aprenderá programación *Backend* y APIs, así como hacer pruebas y depuración de aplicaciones así como su publicación y distribución.

# Diferenciación:

- Incluye curso de nivelación/preparación.
- Certificación internacional con tecnología *Blockchain*, de *Embiz Foundation* (www.embizfoundation.org).
- Certificado de competencias laborales DC-3, de la Secretaría del Trabajo y Previsión Social (STPS).
- Experiencia de Bootcamps, en donde el aspirante se enfrentará a retos prácticos con su profesor y compañeros de grupo con la finalidad de conocer distintas perspectivas y métodos, utilizando herramientas valiosas.
- Valor más alto del mercado y en horarios adecuados para nuestros alumnos.

# Modelo educativo:

El programa estará distribuido de la siguiente manera:

- Acceso a plataforma
- Profesores expertos
- Clases sincrónicas (en vivo)
  - Explicaciones prácticas
  - Ejercicios guiados por el profesor
  - Retos en equipos e individuales
  - Sesiones *live code*, programación (interactiva - en vivo)
- Clases pregrabadas
- Ejercicios
- Foros de interacción
- Casos prácticos
- Exámenes de certificación

# Beneficios:

Al finalizar el programa, el participante obtendrá, además de su diploma de UVM, el Certificado Digital Internacional avalado por *Embiz Foundation*, así como el Certificado DC-3 de Competencias Laborales, avalado por la STPS.

# ¿Qué herramientas de trabajo necesito para cursar un BootCamp en UVM?

1. Contar con computadora de escritorio o laptop y acceso a internet a través de wifi.
2. La computadora debe tener microprocesador Ryzen o Intel i5 o superior, Gen 10 o superior, 8 GB de RAM y 100 GB libres de disco duro.
3. Abrir cuentas para acceso a software libre y en la nube, acceder a un repositorio de documentos, código y proyectos, que el profesor informará en la primera clase.

# TEMARIO

## 01 Introducción al desarrollo de aplicaciones móviles:

**a) Fundamentos de programación:**

1. Variables y tipos de datos:
   - Declaración de variables
   - Tipos de datos primitivos (enteros, flotantes, booleanos, etc.)
   - Uso de constantes y variables mutables
2. Estructuras de control:
   - Condicionales (*If/Else, Switch/Case*)
   - Bucles (*For, While, Do-While*)
   - Uso de operadores lógicos y relacionales
3. Funciones y modularidad:
   - Declaración y llamada de funciones
   - Pasar argumentos y retornar valores
   - Encapsulación de código en funciones

**b) Arquitectura de aplicaciones móviles:**

1. Patrón de diseño MVC (Modelo-Vista-Controlador):
   - Roles y responsabilidades de cada componente
   - Comunicación entre los componentes del MVC
   - Beneficios del patrón MVC en el desarrollo de aplicaciones móviles
2. Comunicación Cliente-Servidor:
   - Protocolo HTTP y REST
   - Métodos HTTP (*GET, POST, PUT, DELETE*)
   - Manipulación de datos en formato JSON
3. Almacenamiento de datos:
   - Bases de datos locales (SQLite, Core Data)
   - Almacenamiento en la nube (Firebase, AWS)
   - Acceso y manipulación de datos en el almacenamiento

**c) Introducción a los sistemas operativos móviles:**

1. 
   - Características principales de iOS:
   - Estructura de una aplicación iOS
   - Ciclo de vida de una app en iOS
   Herramientas de desarrollo
2. (Xcode, Interface Builder)
   - Características principales de Android:
   - Estructura de una aplicación Android
   - Ciclo de vida de una app en Android
   Herramientas de desarrollo (Android Studio, XML)

# 02 Diseño de interfaces de usuario:

**a) Principios de diseño visual:**

1. Teoría del color:
   - Armonía de colores
   - Psicología del color
   - Uso de paletas de colores
2. Tipografía:
   - Selección de fuentes adecuadas
   - Jerarquía de texto
   - Tamaño y espaciado de la tipografía
3. Diseño de íconos y gráficos:
   - Creación de íconos claros y comprensibles
   - Uso de imágenes y gráficos adecuados
   - Consideraciones de tamaño y resolución

**b) Experiencia de Usuario (UX):**

1. Diseño centrado en el usuario:
   - Investigación de usuarios y análisis de necesidades
   - Creación de personas y escenarios de uso
   - Pruebas de usabilidad y retroalimentación del usuario
2. Flujo de interacción:
   - Diseño de *wireframes* y prototipos interactivos
   - Mapas de flujo de la aplicación
   - Retroalimentación y respuesta visual del usuario
3. Diseño adaptable y accesibilidad:
   - Diseño *responsive* para diferentes tamaños de pantalla
   - Consideraciones de accesibilidad para usuarios con discapacidades
   - Uso de estándares de accesibilidad y guías de diseño inclusivas

**c) Herramientas de diseño de interfaces:**

1. Sketch:
   - Creación de diseños de interfaces
   - Uso de símbolos y estilos compartidos
   - Exportación de *assets* y especificaciones de diseño
2. Adobe XD:
   - Diseño de experiencias interactivas
   - Creación de prototipos navegables
   - Diseño colaborativo y compartición de diseños
3. Figma:
   - Diseño de interfaces en la nube
   - Colaboración en tiempo real con equipos de diseño
   - Generación de especificaciones y assets para desarrolladores

# 03 Desarrollo de aplicaciones para iOS:

**a) Introducción a Swift:**

1. Sintaxis avanzada:
   - *Optionals*: manejo de valores nulos
   - *Closures*: funciones anónimas y su uso práctico
   - Control de flujo avanzado: Guard, Defer y Switch avanzado
2. Programación orientada a objetos en Swift: Clases, estructuras y enumeraciones
   - Herencia, polimorfismo y protocolos Propiedades, métodos y propiedades
   - computadas
3. Gestión de memoria:
   - ARC (Automatic Reference Counting)
   - Ciclos de referencia y captura fuerte

**b) Desarrollo de la interfaz de usuario:**

1. Interface Builder:
   - Creación de interfaces visuales utilizando *Storyboards*
   - Conexión de elementos de interfaz a código utilizando IBOutlet e IBAction
   - Gestión de *constraints* y Auto Layout
2. Auto Layout:
   - Uso de *constraints* para definir las relaciones espaciales entre los elementos de interfaz
   - Uso de *stacks* (UIStackView) para simplificar la organización y adaptabilidad de la interfaz
   - Adaptabilidad a diferentes tamaños de pantalla y orientaciones
3. 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

**c) Uso de *frameworks* de iOS:**

1. UIKit:
   - Uso de componentes de interfaz de usuario predefinidos (botones, etiquetas, campos de texto, etcétera)
   - Gestión de eventos de toque y gestos táctiles
   - Uso de animaciones y transiciones de vista
2. 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
3. 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

# 04 Desarrollo de aplicaciones para Android:

**a) Introducción a Kotlin:**

1. 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
2. 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*
3. 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

**b) Creación de interfaces de usuario en Android:**

1. 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
2. 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
3. Recursos y temas personalizados:
   - Creación y uso de recursos: *strings*, dimensiones, colores, estilos
   - Definición de temas personalizados para personalizar la apariencia de la aplicación

**c) Uso de *frameworks* de Android:**

1. 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
2. Room:
   - Configuración de la biblioteca de persistencia Room
   - Creación de entidades y definición de relaciones
   - Uso de consultas y operaciones CRUD con Room DAO
3. 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*

# 05 Programación Backend y APIs:

**a) Introducción a Node.js:**

1. 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

**b) Creación de APIs RESTful:**

1. 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
2. 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

**c) Consumo de APIs en aplicaciones móviles:**

1. Realización de solicitudes HTTP utilizando bibliotecas como Axios (JavaScript) o URLSession (Swift):
   - Configuración de solicitudes con encabezados y datos
   - Manejo de respuestas y errores
2. Manipulación de respuestas en formato JSON:
   - Extracción de datos de respuestas JSON
   - Conversión de datos a objetos utilizables en la aplicación móvil
3. Autenticación y autorización en el consumo de APIs:
   - Uso de *tokens* de autenticación (JWT, OAuth)
   - Protección de rutas y recursos con middleware de autorización

# 06 Pruebas y depuración de aplicaciones:

**a) Pruebas unitarias:**

1. 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
2. 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
3. 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*

**b) Pruebas de interfaz de usuario:**

1. 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
2. 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
3. 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

**c) Depuración de errores:**

1. 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
2. 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
3. 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

# 07 Publicación y distribución de aplicaciones:

**a) Requisitos de envío de la App Store de Apple y Google Play Store:**

1. Preparación de la aplicación para el envío:
   Íconos de la aplicación y pantallas de inicio
   Capturas de pantalla y descripciones de la aplicación
   Requisitos de tamaño de archivo y formatos de imagen
2. Cumplimiento de las directrices de cada tienda:
   Políticas de contenido y restricciones
   Uso de marcas registradas y derechos de autor
   Requisitos de privacidad y seguridad de los datos del usuario
3. Proceso de envío y revisión:
   Creación de una cuenta de desarrollador en cada tienda
   Configuración de la información de la aplicación y envío de la solicitud
   Tiempos de revisión y posibles acciones requeridas

**b) Políticas de revisión y consideraciones de seguridad:**

1. Prácticas recomendadas para pasar la revisión de la tienda de aplicaciones:
   - Diseño y Experiencia de Usuario de alta calidad
   - Cumplimiento de las políticas de contenido y restricciones
   - Calidad y rendimiento de la aplicación
2. Consideraciones de seguridad y privacidad de los datos de los usuarios:
   - Manejo adecuado de datos confidenciales
   - Cumplimiento de regulaciones y estándares de seguridad
   - Implementación de medidas de seguridad, como cifrado y protección de datos

**c) Estrategias de promoción de aplicaciones y análisis de rendimiento:**

1. Técnicas de ASO (App Store Optimization) para mejorar la visibilidad y descargas de la aplicación:
   - Optimización de palabras clave y descripción de la aplicación
   - Obtención de reseñas y calificaciones positivas
   - Aprovechamiento de herramientas de ASO, como Sensor Tower o App Annie
2. Monitoreo y análisis del rendimiento de la aplicación utilizando herramientas como Google Analytics o App Store Connect:
   - Seguimiento de métricas clave, como descargas, usuarios activos y retención
   - Análisis de comportamiento del usuario y flujo de la aplicación
   - Identificación de áreas de mejora y oportunidades de crecimiento
3. Estrategias de *marketing* y promoción para aumentar la adquisición y retención de usuarios:
   - Campañas publicitarias en redes sociales y plataformas de anuncios
   - Colaboraciones con influencers y embajadores de marca
   - Uso de estrategias de retención de usuarios, como programas de fidelidad y notificaciones *push*

# Primer CV corto

**Profesor:**
Alejandro Anaya

**Experiencia Profesional:**

Amplia experiencia de más de 30 años en diversas áreas de Tecnologías de Información, como: Infraestructura, Telecomunicaciones, Seguridad, Desarrollo de Sistemas, Administración, Planeación, Control de Proyectos, Investigación y Desarrollo de Tecnologías, ha trabajado en la formación de equipos multidisciplinarios con el objetivo de integrar grupos resilientes con una amplia autonomía para el trabajo remoto.

Tiene perfecto conocimiento de normas y estándares de las organizaciones que deben cumplir con prácticas de Gobierno Corporativo. Además, ha colaborado en la obtención de certificaciones internacionales (ISO 20000, ISO 9000 e ISO 27000) en organizaciones que deben operar bajo marcos normativos y regulaciones de terceros.

Acostumbrado a la adopción de estándares y la creación de metodologías basadas en Sistemas de Gestión. En el aspecto de inversiones, posee una fuerte disciplina en materia de control de costos sin poner en riesgo la operación; reducción de costos, mediante la implantación de tecnologías basadas en *software* libre; sobre tecnologías licenciadas, supervisión para mantener términos de lo justo necesario.

Experiencia en la construcción de sinergias entre diferentes áreas para llevar a cabo nuevas implantaciones de infraestructura tecnológica, priorizando las necesidades del negocio.

**Formación Académica**

- Más de 500 horas de formación en cursos de especialización práctica.
- Licenciatura en Sistemas de Computación Administrativa, en la Universidad del Valle de México.

**Reconocimientos**

Promociones y logros importantes en diferentes posiciones en el ámbito corporativo.

# Beneficios de estudiar un diplomado con Modelo Educativo Ibaktor

**Obtienes dos certificados:**

- Certificado Internacional digital de alta seguridad y encriptación, con examen de certificación, incluido en el costo de tu diplomado.
- Certificado DC-3 de la STPS.

**Temas actualizados y de vanguardia:**
Con gran capacidad de actualización y reinvención al ser de una duración más corta que otros posgrados, un diplomado te ofrece una capacitación enfocada en temas relevantes y de alta demanda para el mercado laboral.

**Capitaliza lo aprendido:**
El alto enfoque práctico y estratégico de un Diplomado hace que cada módulo sea aplicable desde el primer día 1 en tus actividades profesionales y desarrollo personal.

**Mejora tus oportunidades laborales:**
Enriquece tu CV especializándote y posiciónate como el mejor candidato.

***Networking:***
No solo compartirás salón de clases con buenos compañeros, también con excelentes profesionistas con los que podrás intercambiar puntos de vista, *tips* y oportunidades de negocio.

**Profesores con más 15 años en experiencia profesional:**
Toma clases de la mano de expertos en su disciplina con amplia experiencia compartiendo su conocimiento y trabajando en las mejores empresas nacionales e internacionales.

**Duración:**
La duración del diplomado es de 6 meses, así podrás aplicar lo aprendido muy rápidamente y seguir creciendo profesionalmente.

**Diploma:**
Todos nuestros Diplomados y Certificaciones tienen validez curricular.

# Beneficios del Modelo Educativo Ibaktor

**Clases pregrabadas y en vivo:**
Estudia a tu ritmo, con material de gran calidad, puedes consultar todas las sesiones en el horario que más te convenga. Todas las clases en vivo se graban para tu comodidad.

**Experiencias de aprendizaje *online*:**
Foros.
Juegos.
Ejercicio y herramientas para aplicarlas en tu trabajo o proyectos.
Casos prácticos.
Acceso a materiales complementarios.

**Contenido siempre disponible:**
Podrás consultar y descargar el material desde la plataforma en cualquier momento del día, durante todo el tiempo que dure tu diplomado.
Además, nuestra plataforma es multidispositivo, así podrás estudiar en cualquier computadora de escritorio, *laptop*, tableta o *smartphone*.

**Soporte técnico:**
El equipo de soporte técnico estará a tu disposición en todo momento para ayudarte a resolver cualquier situación.

***Chatbot:***
Mediante el cual te podemos apoyar en todos los temas relacionados con tu experiencia en el diplomado y generamos *tickets* de servicio para tu comodidad, tranquilidad y seguridad.

**Asesoría y acompañamiento:**
Cuentas con Seguimiento Académico a través de *Whatsapp* y otras herramientas a distancia en tiempo real, para resolver tus dudas y dar retroalimentación.

**Evaluación y seguimiento ágil:**
Tendrás retroalimentación fluida y objetiva de tu progreso en el programa para el logro de tu certificado internacional.

UVM / EDUCACIÓN CONTINUA