UNIVERSIDAD TECMILENIO

Guía para el profesor

Desarrollo de Aplicaciones Móviles con Swift

Clave: LSTI2319

Índice

Información general del curso

Modalidades

- Clave banner: LSTI2319
- Modalidad: semestral

Competencia del curso

Desarrolla aplicaciones móviles innovadoras y funcionales para entornos organizacionales, utilizando metodologías ágiles, principios de experiencia de usuario y frameworks de desarrollo multiplataforma, mediante el diseño, implementación y optimización de soluciones tecnológicas en un entorno real o simulado.

Metodología

El modelo académico **MAPS** se caracteriza por ser modular, apilable y personalizable con un enfoque flexible y centrado en el estudiante. Implementamos técnicas didácticas que buscan no solo la adquisición de conocimientos teóricos, sino también la aplicación práctica y el desarrollo de competencias profesionales altamente valoradas por los empleadores. A continuación, se detallan las técnicas didácticas y características principales de nuestro modelo académico.

Técnicas didácticas

Aprendizaje basado en retos. El alumno demuestra la adquisición de los conocimientos y los aplica por medio de retos propuestos.

Aprendizaje basado en proyectos. El alumno demuestra la adquisición de los conocimientos y los aplica en la práctica, por medio de proyectos que impacten de manera positiva a las organizaciones.

Aula invertida*. Esta metodología promueve el autoestudio fuera de las clases, para que, una vez que los alumnos se encuentren en el aula virtual, se promueva la interacción, la construcción conjunta del conocimiento, la generación de ideas y el desarrollo de las competencias, gracias al acompañamiento de docentes expertos.

El **aprendizaje basado en retos** se implementa del primero al quinto semestre, el **aprendizaje basado en proyectos** se aplica del sexto semestre en adelante y la metodología de **aula invertida** está presente en todos los certificados.

*En las Semanas de Desarrollo Integral (SeDI) y los certificados de idioma, solamente aplica la metodología de aula invertida.

Características

1. Certificados
    a. El modelo está formado por certificados de especialidad, los cuales buscan el desarrollo y la adquisición de competencias requeridas por los principales empleadores de nuestro país a través del aprendizaje activo.
    b. Todos los certificados son creados en alianza y colaboración con empresas de prestigio nacional e internacional y/o con expertos que cuentan con conocimiento técnico actual y académico que se requiere en las distintas industrias, con lo que se garantiza el desarrollo de competencias profesionales.
    c. En cada período, el estudiante lleva un máximo de dos certificados simultáneos, con ello los estudiantes tienen la oportunidad de profundizar más en cada tema. Esto es especialmente valioso en cursos que requieren una comprensión detallada de teorías complejas, aplicaciones prácticas y habilidades analíticas avanzadas.

2. Duración
   Licenciatura dependiendo del formato elegido. Los programas ejecutivos se cursan en 15 bimestres, mientras que los programas semestrales se cursan en 8 semestres. Ambos están compuestos porlos mismos certificados en sus mapas curriculares, lo que permite transitar entre ambas modalidades dependiendo de las necesidades de los estudiantes.

3. Flexibilidad
   Este modelo promueve la participación de los estudiantes al permitirles personalizar su experiencia de aprendizaje de acuerdo con sus intereses y necesidades individuales. Esta personalización no solo facilita un mayor compromiso y motivación, sino que también prepara a los estudiantes para enfrentar retos específicos de su futuro campo profesional, aumentando así su empleabilidad y éxito académico.

4. Credenciales apilables
   La idea atrás de estas credenciales es proveer un esquema de capacitación y aprendizaje para los aprendedores, de tal forma que puedan moverse rápidamente en el proceso educativo,

aprendiendo habilidades que son aplicables en el trabajo. Las credenciales, por lo tanto, pueden ser apiladas para cumplir con el estándar de un programa de grado tradicional.

5. Insignias digitales
   Las insignias digitales permiten documentar la educación de los estudiantes, así como sus logros. Una de las ventajas de las insignias digitales es que, a través de la metadata, se pueden obtener los detalles de las competencias adquiridas, la institución que otorga la insignia, así como un reconocimiento visual que puede ser compartido en redes sociales o redes profesionales.

6. Diferenciadores del modelo

   a. Certificados de lengua extranjera: se cuenta con certificados para adquirir o reforzar el dominio de lengua extranjera y con certificados impartidos en una lengua extranjera específicos de la disciplina, todo con el objetivo de atender las demandas de los empleadores.
   b. Semanas de Desarrollo Integral: unidades de aprendizaje transversal, diseñadas para vivir una experiencia inmersiva, desarrollando las competencias humanas, profesionales y de bienestar.
   c. Períodos de Skilling: período complementario donde el alumno puede llevar a cabo actividades que suman a su formación académica. Son opcionales y personalizadas, ya que el estudiante las selecciona con base en sus intereses profesionales y personales.
   d. Estancia empresarial al final del programa de estudios: los estudiantes tendrán a su disposición tres opciones en función de la estancia empresarial que vayan a realizar, entre las cuales se encuentran: gestión de proyectos, emprendimiento y desarrollo sostenible.

Evaluación

Elemento	Evaluables	Puntos
1	Actividad 1	6
2	Actividad 2	6
3	Avance 1 del proyecto	25
4	Actividad 3	6
5	Actividad 4	6
6	Actividad 5	6
7	Entrega proyecto final	35
8	Presentación de reto/proyecto o examen o feria de retos y proyectos	10
	Total	**100**

Bibliografía

Libro de texto

Apple Education. (2024). *Develop in Swift Explorations*. Recuperado de https://books.apple.com/us/book/develop-in-swift-explorations/id1581182728

Apple Education. (2024). *Develop in Swift Fundamentals*. Recuperado de https://books.apple.com/us/book/develop-in-swift-fundamentals/id6468967906

Apple Inc. (2020). *Design Workbook: Designing Great Apps*. Recuperado de https://developer.apple.com/design/

Apple Inc. (2020). *Designing for iOS*. Recuperado de https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/

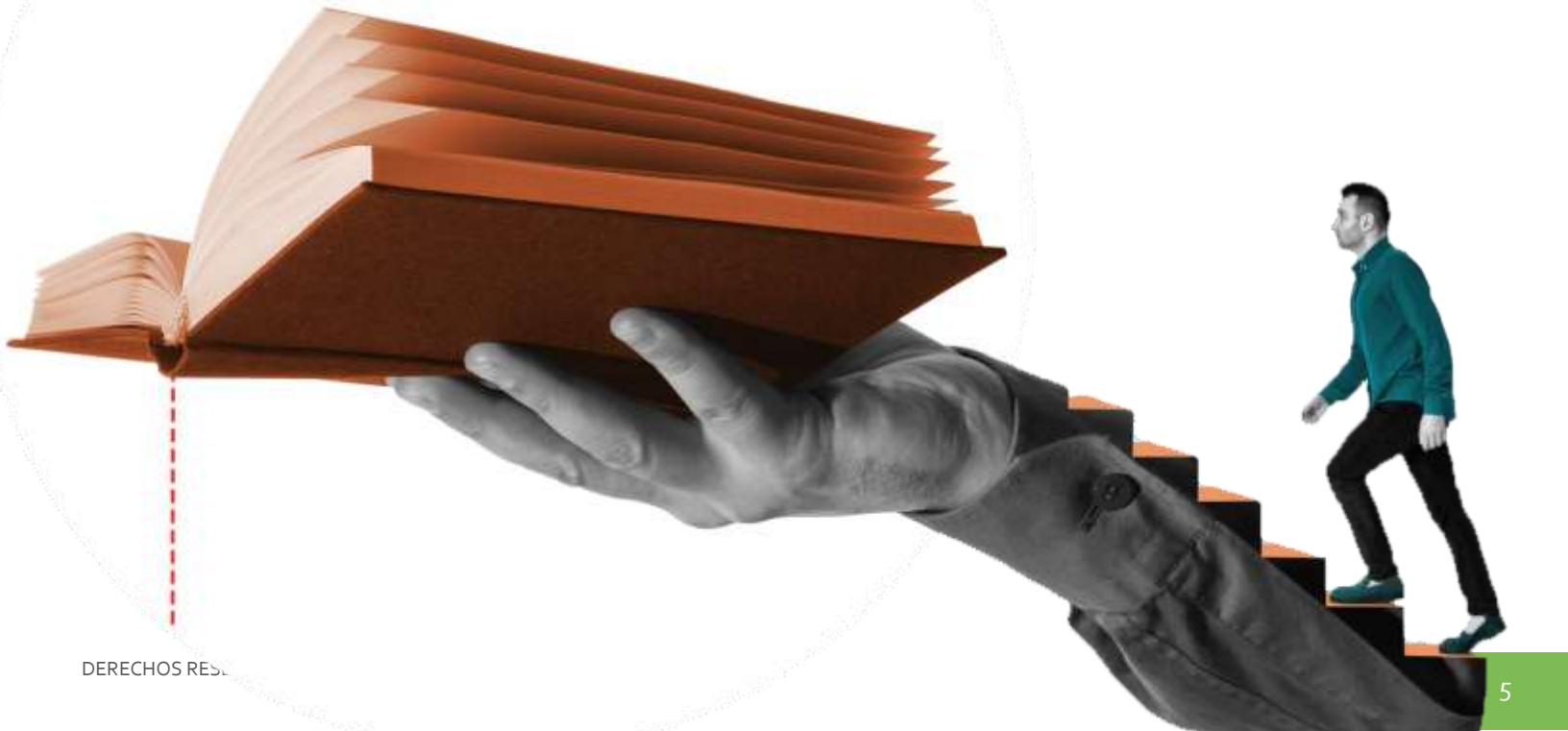
Apple Inc. (2020). *Develop in Swift. App Design Workbook*. Recuperado de https://www.apple.com/au/education/docs/app-design-workbook-AU.pdf

Apple Inc. (2024). *Develop in Swift Curriculum Guide: Xcode 15*. Recuperado de https://atlc.apple.com/downloads/xcode15/Develop_in_Swift_Curriculum_Guide_Xcode_15_Feb2024.pdf

Libro de apoyo

Keur, C., y Hillegass, A. (2021). *iOS programming: The Big Nerd Ranch guide* (8ª ed.). Big Nerd Ranch.

Lee, W. (2020). *SwiftUI for dummies*. A Wiley Brand.



Tips importantes

Material de capacitación en la plataforma tecnológica Canvas

- Tutorial digital para profesores:
- Tutorial digital para alumnos:

- **¿En dónde o a quién reporto un error detectado en el contenido del curso?**

Lo puedes reportar a la cuenta atencioncursos@servicios.tecmilenio.mx, pero te pedimos que también reportes sugerencias para el contenido y actividades del curso.

- **¿Quién me informa de la cantidad de sesiones y tiempo de cada sesión en las semanas?**
- **¿Tengo que capturar las calificaciones en banner y en la plataforma educativa?**

Sí, es importante que captures calificaciones en la plataforma para que los alumnos estén informados de su avance y reciban retroalimentación de parte tuya de todo lo que realizan en el curso. En banner es el registro oficial de las calificaciones de los alumnos.

Temario

1	**Panorama general del desarrollo de aplicaciones móviles en iOS**
1.1	Definición de aplicaciones móviles y su relevancia en el entorno organizacional
1.2	Tipos de aplicaciones móviles: nativas y multiplataforma
1.3	Introducción a iOS, Swift y el ecosistema de Apple
2	Herramientas de desarrollo y configuración del entorno
2.1	Instalación de Xcode y configuración del entorno de desarrollo
2.2	Creación de un primer proyecto en Xcode
2.3	Familiarización con Swift y sus principales características
2.4	Primera aplicación en Swift
3	Fundamentos del Diseño Centrado en el Usuario (UX)
3.1	Principios básicos del diseño UX: usabilidad, accesibilidad y emocionalidad
3.2	Técnicas de investigación de usuarios: encuestas, entrevistas y pruebas de usabilidad
3.3	Creación de personas (perfiles de usuario) y análisis de necesidades
3.4	Diseño iterativo y prototipos rápidos

3.5	Playground
4	Herramientas y técnicas de diseño para aplicaciones móviles
4.1	Introducción a las herramientas de diseño
4.2	Creación de wireframes y prototipos de baja fidelidad
4.3	Diseño de la estructura de navegación y flujos de usuario
4.4	Establecimiento de jerarquías visuales y accesibilidad en interfaces móviles
5	Principios de Diseño de Interfaces de Usuario (UI) en iOS
5.1	Elementos de la interfaz móvil en iOS: botones, formularios y menús
5.2	Uso de color, tipografía, iconos y espacio en blanco en interfaces
5.3	Diseño visual coherente y consistencia en la experiencia de usuario
5.4	Patrones de diseño y guías de estilo para iOS
6	Diseño avanzado de interfaces con SwiftUI
6.1	Introducción a SwiftUI para la creación de UI en iOS
6.2	Creación de vistas, formularios y navegación con SwiftUI
6.3	Manejo de datos dinámicos y actualización de vistas en tiempo real
6.4	Implementación de temas y personalización de la interfaz
7	Manejo de eventos y entradas del usuario en iOS
7.1	Interacciones básicas: clics, gestos y multitouch
7.2	Formularios y validaciones de entrada
7.3	Gestión de eventos y respuestas a interacciones del usuario
8	Integración de bases de datos locales y persistencia de datos
8.1	Uso de Core Data para la persistencia de datos locales en iOS
8.2	Creación de entidades y relaciones en Core Data
8.3	Interacción con bases de datos a través de Swift
8.4	Diseño de pantallas que visualizan y gestionan datos de forma intuitiva
9	Servicios en la nube

9.1	Introducción a Firebase para iOS: autenticación y base de datos en tiempo real
9.2	Conexión y sincronización de datos con Firebase
9.3	Autenticación de usuarios con Firebase Authentication
10	Conceptos básicos de aprendizaje automático y realidad aumentada
10.1	Proceso de vida entrenamiento, prueba y perfeccionamiento de Co-ML
10.2	Creación de una app de aprendizaje automática contexto y diseño
10.3	Configuración del entorno de desarrollo con ARKit
10.4	Introducción al ARKit y RealityKit
10.5	Primer proyecto con ARKit en SwiftUI
11	Accesibilidad y diseño inclusivo en aplicaciones móviles
11.1	Fundamentos de accesibilidad en iOS: compatibilidad con VoiceOver y otros accesos
11.2	Mejores prácticas para crear aplicaciones accesibles
11.3	Diseño para personas con discapacidades visuales, auditivas y motoras
11.4	Herramientas de prueba de accesibilidad en Xcode
12	Prácticas de accesibilidad en iOS
12.1	Optimización de interfaces para diferentes tamaños y resoluciones de pantalla
12.2	Diseño responsivo para distintos dispositivos iOS (iPhone, iPad y Apple Watch)
12.3	Pruebas y ajustes para garantizar la accesibilidad
12.4	Gestión de color y contraste para usuarios con deficiencias visuales
13	Gestión del ciclo de vida de aplicaciones móviles
13.1	Estrategias para la gestión de versiones y actualizaciones de aplicaciones
13.2	Uso de control de versiones con Git para diseñadores y desarrolladores
13.3	Técnicas de pruebas: unitarias y de integración en Xcode
13.4	Manejo de excepciones y errores en aplicaciones móviles
14	Despliegue y publicación de aplicaciones en App Store
14.1	Proceso de publicación en la App Store
14.2	Requisitos técnicos, políticas y optimización para la publicación

14.3	Creación de metadatos, capturas de pantalla y descripciones atractivas
14.4	Estrategias de mantenimiento post-lanzamiento y actualizaciones

Notas de enseñanza

Tema 1 **Panorama general del desarrollo de aplicaciones móviles en IOS**

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Inicie la sesión conectando el contenido con el entorno actual.
   Contextualícese la importancia de las aplicaciones móviles en la vida cotidiana y en sectores como salud, educación, comercio y movilidad. Enfáticamente se debe mostrar cómo iOS ha transformado la forma en que se interactúa con la tecnología, desde la perspectiva del usuario y del desarrollador.
2. Explique qué es una aplicación móvil dentro del ecosistema iOS.
   Preséntese su definición, resáltese sus características clave (desempeño, accesibilidad, integración con el hardware de Apple) y descríbase cómo se diferencia del desarrollo en otras plataformas. Empléense ejemplos como Mail, Notas, Medidas o Safari para ilustrar el uso eficiente del hardware.
3. Presente el enfoque de Apple hacia la experiencia de usuario.
   Súbrayese el rol de las *Human Interface Guidelines*, la integración de tecnologías como Face ID, ARKit, CoreML y HealthKit. Analícese la importancia de la privacidad, la seguridad y la accesibilidad como principios de diseño.
4. Use la historia de Apple como un recurso narrativo.
   Relátese brevemente la evolución de Apple desde el garaje de Steve Jobs hasta convertirse en líder tecnológico global. Relaciónese esta historia con el surgimiento del iPhone, la App Store y el impacto del desarrollo en iOS.
5. Introduzca SwiftUI como herramienta moderna de desarrollo.
   Describa su enfoque declarativo, su integración con Xcode, las ventajas en productividad, escalabilidad y compatibilidad multiplataforma (iOS, macOS, watchOS, tvOS, visionOS). Destáquese su importancia en el ámbito educativo y profesional.
6. Compare tipos de aplicaciones: nativas vs. multiplataforma.
   Incorpórese una tabla o diagrama (SmartArt) con sus diferencias en desempeño, experiencia de usuario, acceso al hardware y tiempo de desarrollo. Inclúyanse ejemplos de tecnologías como SwiftUI, React Native, Flutter, Xamarin. Foméntese la reflexión sobre cuándo conviene usar cada enfoque.
7. Analice casos de éxito en iOS.
   Expónganse y discútanse los cinco ejemplos incluidos en el material (Airbnb, Nike, IKEA, Duolingo, Tesla). Asígnese a equipos analizar qué tecnologías iOS utilizó cada empresa, qué problemas resolvió y qué impacto obtuvo. Impúlsese el debate sobre el valor estratégico del desarrollo móvil.

8. Conecte con la evolución de iOS y su impacto en la experiencia de usuario.
   Explíquense los hitos del sistema operativo: iOS 1 hasta iOS 18, aparición del App Store, Siri, Widgets, ARKit, NameDrop, integración de IA generativa. Solicítese al grupo identificar qué cambios han vivido como usuarios y cómo influyen en el diseño de apps.
9. Cierre con una reflexión guiada.
   Propónganse preguntas como: ¿Qué tipo de aplicación desarrollarías tú y por qué? ¿Qué elementos influyen más en una buena experiencia de usuario? ¿Cómo puede una app mejorar la vida de las personas? Estimúlense respuestas vinculadas a necesidades reales.
10. Recomiende recursos complementarios.
    Invítese a consultar los videos, lecturas y pódcast del material. Asígnense como tarea o como parte de una actividad de reflexión sobre tendencias emergentes como Vision Pro, Apple Silicon, inteligencia artificial o realidad aumentada.

Tema 2 Herramientas de desarrollo y configuración del entorno

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Inicie la sesión resaltando la importancia de un entorno de desarrollo profesional.

Explíquese cómo el dominio de herramientas como Xcode y Swift resulta fundamental para transformar ideas en aplicaciones funcionales dentro del ecosistema Apple. Relaciónese con ejemplos cotidianos de aplicaciones desarrolladas en iOS y menciónese que este será el primer acercamiento práctico del curso.

2. Presente Xcode como el entorno de desarrollo oficial de Apple.

Describa su propósito, historia, evolución y funciones clave. Enfóquese en sus herramientas integradas: editor de código, simuladores, Interface Builder, pruebas automatizadas y soporte para múltiples plataformas Apple. Reprodúzcase en clase la instalación guiada desde la Mac App Store y muéstrese la interfaz de Xcode en vivo si resulta posible.

3. Guíe paso a paso la instalación de Xcode 16.

Verifíquese que los estudiantes conozcan los requisitos previos y sigan de forma adecuada el proceso desde la App Store. Expónganse las pantallas clave del proceso y aprovéchese para enseñar atajos útiles de teclado desde el inicio. Oriéntese sobre cómo personalizar la apariencia de Xcode (temas, fuentes, vista clara u oscura).

4. Facilite la creación del primer proyecto en Xcode.

Acompáñese a los estudiantes en el proceso de crear un nuevo proyecto en SwiftUI, explíquese la estructura básica del mismo (App.swift, ContentView.swift, Assets) y ejecútese la aplicación en el simulador. Asígnese una práctica guiada, como una aplicación de frases motivacionales.

5. Presente Swift como un lenguaje moderno y seguro.

Describan sus principales ventajas: claridad, desempeño, manejo de errores, seguridad de tipos, inferencia y expresividad. Compárese brevemente con otros lenguajes conocidos por los estudiantes (Python, Java, entre otros). Muestren ejemplos básicos en el proyector o en Playgrounds.

6. Enseñe los fundamentos de programación en Swift.

Introduzcan variables, constantes, tipos de datos, operadores, inferencia y anotación de tipos. Destáquese la diferencia entre let y var, cómo evitar errores frecuentes y la importancia de escribir código claro y comprensible.

7. Explore la estructura básica de un programa con SwiftUI.

Explíquense los conceptos de struct, body, jerarquía de vistas (VStack, HStack, ZStack), modificadores y previsualizaciones (Preview). Muestre cómo crear interfaces visuales con pocas líneas de código. Foméntese la exploración libre en Playgrounds o en ejercicios breves durante la clase.

8. Cierre con una reflexión guiada.

Invítese al grupo a reflexionar: ¿cómo se relaciona un entorno bien configurado con el éxito de una aplicación? ¿Qué ventajas se identifican en Swift en comparación con otros lenguajes? ¿Cómo influye la claridad del código en la calidad del producto final?

Tema 3 Fundamentos del Diseño Centrado en el Usuario (UX)

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Inicie la sesión presentando Swift como una herramienta poderosa y accesible.

Describa Swift como el lenguaje de programación oficial de Apple, moderno, rápido y seguro. Resáltese cómo permite crear aplicaciones para iPhone, iPad, Apple Watch, Mac y Vision Pro. Menciónese que su sintaxis limpia y sus mecanismos de seguridad lo vuelven ideal tanto para principiantes como para desarrolladores con experiencia.

2. Establezca el propósito del tema: comprender la lógica del lenguaje Swift.

Indíquese que dominar los fundamentos de Swift resulta esencial para crear aplicaciones funcionales, estables y eficientes. Enfóquese en que este conocimiento servirá como base para todos los desarrollos posteriores en el curso.

3. Explique las diferencias entre variables y constantes.

Aclárese el uso de *var* (variables que pueden cambiar su valor) y *let* (constantes inmutables). Promuévanse buenas prácticas, como utilizar *let* siempre que sea posible, para optimizar el rendimiento y prevenir errores. Realícense ejercicios prácticos para que el estudiante experimente el cambio y la inmutabilidad.

4. Introduzca los tipos de datos básicos de Swift.

Enumérense y explíquense *Int*, *Double*, *Float*, *String* y *Bool*. Destaque la importancia del tipado fuerte en Swift y sus beneficios para evitar errores durante la ejecución. Empléense ejemplos concretos y visuales para ilustrar las diferencias.

5. Enseñe la inferencia y anotación de tipos.

Muestre cómo Swift puede deducir el tipo de dato de forma automática (inferencia) y cuándo resulta necesario especificarlo (anotación). Preséntense ejemplos que contrasten ambos enfoques y subráyese su utilidad para mantener un código claro y robusto.

6. Explore los operadores fundamentales.

Trabájese con operadores aritméticos, de asignación y lógicos. Propónganse ejercicios breves que combinen

operadores y tipos de datos para practicar la sintaxis. Aprovéchese la oportunidad para introducir nociones de expresiones booleanas simples y estructuras condicionales.
7. Presente la estructura general de un programa en Swift.
Explíquese la importación de módulos (*import SwiftUI*), la definición de estructuras (*struct*), la propiedad *body* y la jerarquía de vistas. Relaciónese esto con la estructura de una aplicación creada en Xcode y cómo estas piezas se integran entre sí.
8. Demuestre el uso de modificadores y jerarquía visual en SwiftUI.
Ilústrense ejemplos de cómo los elementos visuales (*Text*, *Image*, *Button*) se organizan mediante *VStack*, *HStack* y *ZStack*, y cómo se les aplican modificadores (*font*, *foregroundColor*, *padding*, entre otros) para darles estilo. Invítese al alumnado a modificar vistas básicas y observar los resultados en tiempo real.
9. Realice una práctica guiada con una aplicación simple.
Utilícese el ejemplo de una aplicación de frases motivacionales. Explíquese paso a paso la creación, ejecución y modificación del código en *ContentView.swift*. Anímese a los estudiantes a personalizar frases, colores o el diseño visual.
10. Cierre con una reflexión guiada.
Formúlese preguntas como: ¿Qué ventajas ofrece la claridad de Swift? ¿Cómo facilita la inferencia de tipos el proceso de programación? ¿Por qué resulta importante comprender bien los tipos de datos y operadores antes de avanzar?

Tema 4 Herramientas y técnicas de diseño para aplicaciones móviles

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Inicie la sesión explicando la importancia del diseño centrado en el usuario en aplicaciones móviles.
Destáquese cómo la competencia y las expectativas de los usuarios exigen experiencias memorables y funcionales, y preséntense ejemplos de aplicaciones exitosas que aplican estos principios. Relaciónese el impacto directo en la retención y satisfacción del usuario.
2. Explique los principios básicos del diseño UX: usabilidad, accesibilidad y emocionalidad.
Desglósense los componentes clave de la usabilidad según Nielsen (aprendizaje, eficiencia, memorabilidad, manejo de errores y satisfacción) y relaciónense con ejemplos prácticos en Android y iOS. Profundícese en los principios de accesibilidad de la WCAG 3.0 (perceptible, operable, comprensible y robusto) y destáquese la importancia de la emocionalidad en la conexión entre usuario y aplicación.
3. Demuestre cómo aplicar los principios de usabilidad y accesibilidad en interfaces móviles.
Realícense ejemplos prácticos en clase, como diseñar botones accesibles en XML para Android, ajustar contrastes y tamaños de texto, y habilitar etiquetas *contentDescription*. Invítese a los estudiantes a utilizar herramientas como *Android Accessibility Scanner* para validar sus diseños.
4. Presente y compare técnicas de investigación de usuarios: encuestas, entrevistas y pruebas de usabilidad.
Explíquese cuándo y cómo aplicar cada método, inclúyanse ejemplos de preguntas eficaces y tareas para pruebas. Recomiéndense plataformas como Google Forms, Typeform y Maze, y enfáticamente resáltese la importancia de observar y analizar el comportamiento real de los usuarios antes de tomar decisiones de diseño.

5. Guíe la creación de personas y el análisis de necesidades.
Explíquese cómo recolectar datos (entrevistas, encuestas, análisis de métricas), identificar patrones y construir arquetipos de usuario. Realícese un ejercicio práctico de creación de una persona y priorización de necesidades funcionales, sociales y emocionales, utilizando herramientas como mapas de empatía y matrices de impacto.
6. Explique el proceso de diseño iterativo y prototipos rápidos.
Descríbase el ciclo ideación → prototipo de baja fidelidad → pruebas con usuarios → análisis, y muéstrese cómo este enfoque reduce riesgos y acelera la mejora del producto. Propóngase la regla 5-5-5 (máximo cinco pantallas, cinco días, cinco usuarios para pruebas) y el uso de herramientas como Figma y ProtoPie para prototipado rápido.

Tema 5 Principios de Diseño de Interfaces de Usuario (UI) en iOS

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Comience la sesión contextualizando la importancia del diseño en la retención y éxito de aplicaciones móviles.
   Explíquese que, según datos recientes, el 68 % de las aplicaciones se desinstalan durante la primera semana por problemas de usabilidad, navegación confusa o interfaces inaccesibles. Relaciónese cómo el dominio de herramientas y técnicas de diseño representa hoy una necesidad estratégica y no únicamente estética.
2. Presente y compare las principales herramientas de diseño colaborativo utilizadas en la industria.
   Demuestren sus ventajas Figma (colaboración en tiempo real, complementos), Adobe XD (integración con el ecosistema Adobe), Sketch (diseño vectorial en Mac), InVision y Proto.io (prototipado interactivo). Inclúyase una breve demostración práctica de Figma y resáltese el papel de herramientas potenciadas por inteligencia artificial como Uizard y Adobe Sensei para acelerar la iteración de prototipos.
3. Explique y guíe la creación de *wireframes* y prototipos de baja fidelidad.
   Descríbase la metodología en cuatro pasos: investigación y definición de requerimientos, bocetado manual o digital, prototipado interactivo y validación con usuarios. Realícese un ejercicio práctico de bocetado rápido y muéstrense ejemplos de *wireframes* efectivos, enfatizándose la importancia de identificar problemas de usabilidad antes de invertir en desarrollo.
4. Describa el diseño de la estructura de navegación y los flujos de usuario.
   Analícense patrones de navegación móvil como barras de pestañas (*tab bars*), menús tipo hamburguesa, navegación por gestos y asistentes (*wizard*). Explíquese cómo definir tareas críticas, segmentar usuarios y crear *wireflows* que incluyan estados de error y rutas alternativas. Recomiéndese probar los flujos con usuarios reales y utilizar métricas como tasa de éxito y *System Usability Scale* para validar la experiencia.
5. Demuestre la aplicación de principios de jerarquía visual en interfaces móviles.
   Ilústrese cómo el contraste, el color, el tamaño, la tipografía, el espaciado y la profundidad guían la atención del usuario y facilitan la toma de decisiones. Realícese un análisis de casos prácticos y

propónganse ejercicios de rediseño aplicando los principios de la Gestalt y Material Design para mejorar la claridad y el enfoque de las pantallas.

6. Enfatice la importancia de la accesibilidad en el diseño móvil.
   Explíquense los principios clave: contraste adecuado, tamaño mínimo de elementos, etiquetas descriptivas, navegación simplificada y pruebas con personas con discapacidad. Muéstrese cómo implementar *contentDescription* en Android y cómo validar el cumplimiento de estándares WCAG y UNE-EN 301 549:2019. Propóngase la integración de opciones de personalización visual y el uso de lectores de pantalla durante las pruebas.

Tema 6 Diseño avanzado de interfaces con SwiftUI

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Inicie la sesión resaltando la evolución del diseño de interfaces en iOS.
   Explíquese cómo el enfoque declarativo de SwiftUI ha transformado el desarrollo visual en el ecosistema Apple. Utilícese una breve historia comparativa entre *UIKit* y *SwiftUI* para contextualizar el avance técnico y pedagógico.
2. Presente las ventajas de usar SwiftUI para diseño de interfaces.
   Subráyese su claridad sintáctica, compatibilidad con todas las plataformas Apple y soporte nativo para accesibilidad, temas visuales, navegación y reactividad. Menciónese que se trata del estándar actual y futuro para desarrollos profesionales en iOS.
3. Demuestre el uso de vistas básicas y contenedores.
   Explíquense las vistas fundamentales como *Text*, *Image*, *Button*, *Form*, *List* y cómo se estructuran con *VStack*, *HStack* y *ZStack*. Preséntense ejemplos breves en Xcode y motívese al grupo a experimentar con estas combinaciones desde el inicio.
4. Enseñe la navegación estructurada con *NavigationStack*.
   Compárese *NavigationStack* con *NavigationView* y muéstrese cómo permite flujos más seguros y tipados. Genérese una navegación simple entre dos pantallas para ilustrar el uso de *NavigationLink* y *NavigationPath*.
5. Explique el manejo de datos en tiempo real con SwiftUI.
   Introdúzcanse los *property wrappers* (*@State*, *@Binding*, *@ObservedObject*, *@Environment*, *@EnvironmentObject*) mediante ejemplos cortos e interactivos. Enfóquese en cómo estos mecanismos permiten que la vista se actualice de forma dinámica sin recarga manual.
6. Aborde la personalización visual y los temas.
   Preséntese cómo implementar modo oscuro y claro usando el *Asset Catalog* y colores del sistema. Explíquense modificadores como *.tint()*, *.background()*, *.cornerRadius()* y cómo estructurar temas globales mediante *enum*, *singleton* y *@EnvironmentObject*.
7. Refuerce la importancia de la accesibilidad desde el diseño.
   Coméntese el uso de *.accessibilityLabel*, *.accessibilityHint*, *Dynamic Type*, alto contraste y navegación

por *VoiceOver*. Resáltese que SwiftUI incorpora accesibilidad desde su diseño e invítese al grupo a probar sus aplicaciones con estas herramientas.

8. Cierre con una reflexión guiada.
   Propónganse preguntas como: ¿Cómo ayuda SwiftUI a escribir menos código para lograr más? ¿Qué ventaja ofrece trabajar con datos reactivos? ¿Cómo influye la accesibilidad en la experiencia final del usuario?

Tema 7 Manejo de eventos y entradas del usuario en iOS

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Inicie la sesión contextualizando el valor de la interacción entre el usuario y la aplicación.
   Explíquese cómo la respuesta a toques, gestos e ingreso de datos convierte una aplicación en una experiencia interactiva y significativa. Menciónense ejemplos comunes: deslizar para eliminar, tocar para seleccionar, mantener presionado para abrir menús.
2. Presente el concepto de evento y su importancia en el desarrollo móvil.
   Coméntese que los eventos permiten que las aplicaciones “escuchen” e interpreten las acciones del usuario. Indíquese que, en iOS, estos eventos se gestionan de forma distinta según se utilice SwiftUI o UIKit, pero el objetivo siempre consiste en capturar la intención del usuario y responder con fluidez.
3. Explique las interacciones básicas: clics, gestos y multitouch.
   Utilícense ejemplos prácticos en Xcode para mostrar *onTapGesture*, *onLongPressGesture* y gestos multitáctiles como *pinch* o *rotate*. Compárese la simplicidad de SwiftUI frente al control detallado de UIKit con métodos como *touchesBegan* o *UIGestureRecognizer*.
4. Analice cómo se diseñan formularios en SwiftUI y su validación.
   Demuéstrese la creación de formularios con *Form*, *TextField*, *Picker*, *Toggle*, y cómo validar entradas utilizando *@State* y *onChange*. Enfáticamente, resáltese la importancia de proporcionar retroalimentación inmediata y accesible para el usuario final.
5. Presente la arquitectura de eventos en UIKit.
   Explíquese brevemente el ciclo de captura, distribución y respuesta con la *responder chain* (*UIResponder*). Muéstrese cómo personalizar la detección de gestos complejos y cuándo conviene usar reconocedores (*UIGestureRecognizer*) en lugar de detección directa.
6. Aborde la gestión declarativa de eventos con SwiftUI.
   Listénse y demuéstrense modificadores clave como *.onTapGesture*, *.onSubmit*, *.onReceive* y *.onChange*. Explíquese cómo se emplean para responder a cambios en *@State*, acciones del usuario o eventos del sistema, como la aparición del teclado o el cambio de orientación.
7. Hable sobre la accesibilidad aplicada a eventos.
   Preséntese cómo agregar *UIAccessibilityCustomAction* y utilizar notificaciones del sistema para adaptar la interfaz a personas con discapacidad. Recuérdese al estudiantado que la accesibilidad también forma parte del diseño de calidad.

8. Muestre cómo reforzar la interacción con retroalimentación visual y háptica.
   Realícese una demostración con *UIImpactFeedbackGenerator* y animaciones con *withAnimation {}* para mostrar cómo estas respuestas mejoran la experiencia y la percepción del usuario.
9. Cierre con una reflexión guiada.
   Propónganse preguntas como: ¿Qué tipo de interacción resulta más intuitiva para el usuario? ¿Cuándo conviene usar gestos simples o compuestos? ¿Cómo puede lograrse una aplicación más accesible desde el enfoque de la interacción?

Tema 8 Integración de bases de datos locales y persistencia de datos

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Inicie la sesión resaltando la importancia de la persistencia de datos en aplicaciones móviles.
   Explíquese que una aplicación que almacena información de forma local puede funcionar sin conexión, conservar el estado del usuario y mejorar la experiencia general. Enfáticamente, destáquese que *Core Data* es la herramienta nativa de Apple para lograr esta funcionalidad de manera estructurada, eficiente y segura.
2. Presente qué es *Core Data* y por qué se utiliza.
   Coméntese que *Core Data* es un *framework* orientado a objetos que permite gestionar datos sin necesidad de escribir SQL de forma directa. Explíquense sus beneficios: buen rendimiento, escalabilidad, integración con SwiftUI y manejo automático de relaciones.
3. Explique los componentes principales de *Core Data*.
   Preséntese visualmente la *Core Data Stack*: *NSManagedObjectModel*, *NSManagedObjectContext*, *NSPersistentContainer*, y descríbase cómo trabajan en conjunto. Utilícense gráficos *SmartArt* o diagramas para facilitar la comprensión del flujo de datos.
4. Demuestre la creación de entidades y relaciones en Xcode.
   Guíese al grupo en la creación de un modelo de datos (*.xcdatamodeld*) con entidades como Usuario, Proyecto y Tarea. Explíquese cómo definir atributos, establecer relaciones (1:1, 1:N, N:M) y configurar reglas de borrado (*Cascade*, *Nullify*, *Deny*).
5. Enseñe cómo realizar operaciones CRUD con Swift.
   Muéstrese en código cómo insertar, consultar (*NSFetchRequest*), actualizar y eliminar objetos en *Core Data*. Resáltese el uso de buenas prácticas como el manejo de errores, el empleo de contextos en segundo plano y la validación de datos.
6. Integre la interfaz de usuario con *Core Data* en SwiftUI.
   Explíquese el uso de *@FetchRequest* para mostrar listas dinámicas conectadas a los datos, *@Environment(.managedObjectContext)* para guardar cambios, y cómo crear formularios vinculados directamente con objetos observables.
7. Analice la importancia de una interfaz intuitiva al gestionar datos.
   Muéstrese cómo diseñar pantallas donde el usuario puede ver, crear y editar datos con facilidad.

Enfáticamente, subráyese la necesidad de usar listas reactivas, formularios accesibles y retroalimentación visual (alertas, validaciones, confirmaciones).

8. Cierre con una reflexión guiada.
   Propónganse preguntas como: ¿Cuándo se justifica utilizar *Core Data* en lugar de otras formas de almacenamiento? ¿Qué retos se han encontrado al trabajar con relaciones entre datos? ¿Cómo impacta la persistencia de datos en la experiencia del usuario?

Tema 9 Servicios en la nube

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Comience la sesión contextualizando la importancia de la optimización de la interacción en la retención de usuarios.
2. Explíquese que, en un mercado saturado de aplicaciones, la fluidez, adaptabilidad y precisión de las interacciones son factores críticos para evitar desinstalaciones tempranas y destacar frente a la competencia. Preséntense estadísticas recientes sobre la tasa de abandono y la preferencia de los usuarios por aplicaciones que integran animaciones y gestos bien implementados.
3. Explique y demuestre la creación de animaciones y transiciones fluidas en Android.
   Destáquese que las animaciones no solo embellecen, sino que guían la atención, refuerzan acciones y mejoran la percepción del rendimiento. Muéstrense ejemplos prácticos con Jetpack Compose (*animate*AsState*, transiciones de *layouts*) y discútanse los principios de duración, suavizado y consistencia según *Material Design 3.0*. Realícense ejercicios de animaciones de retroalimentación y transiciones entre pantallas.
4. Presente la implementación de gestos y microinteracciones como lenguaje de interacción moderno.
   Explíquese la diferencia entre *GestureDetector* y los modificadores de gestos en Compose. Realícense demostraciones de gestos estándar (deslizar, toque largo, pellizcar) y microinteracciones visuales y hápticas (*ripple effect*, vibración). Introdúzcase el uso de *MediaPipe* para reconocimiento avanzado de gestos y expónganse buenas prácticas para evitar fatiga y errores de interpretación.
5. Guíe a los estudiantes en la mejora de la experiencia táctil y la interacción en dispositivos móviles.
   Explíquese la importancia de respetar tamaños mínimos de elementos interactivos (≥48 dp), mantener espaciado adecuado y ofrecer retroalimentación inmediata. Realícense ejercicios para implementar zonas seguras, control de repetición de eventos (*debounce*) y prevención de toques accidentales. Analícese cómo calibrar la sensibilidad táctil y adaptar la experiencia a dispositivos con protectores de pantalla o requerimientos especiales.
6. Explique el diseño adaptativo para dispositivos con pantallas pequeñas y grandes.
   Analícense los principios *mobile-first*, la priorización del contenido esencial y la jerarquía visual clara en móviles. Demuéstrese el uso de *multi-pane layouts*, *Navigation Rail* y contenido adaptable en tabletas y dispositivos plegables. Realícense ejercicios prácticos con *ConstraintLayout* y *WindowSizeClass* para

validar la adaptabilidad de la interfaz y la optimización de recursos gráficos (*VectorDrawables*, *WebP*, *Lazy Loading*).

7. Proponga actividades prácticas de revisión y optimización de la interacción.
   Invítese a los estudiantes a validar la duración y el suavizado de animaciones, implementar gestos estándar con retroalimentación, verificar el tamaño y espaciado de elementos táctiles, y probar diseños responsivos en múltiples dispositivos. Sugiérase el uso de herramientas como *Android Accessibility Scanner* y *Color Contrast Checker* para asegurar accesibilidad y calidad.

Tema 10 Conceptos básicos de aprendizaje automático y realidad aumentada

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Inicie la sesión con un caso de uso impactante.
   Preséntese cómo aplicaciones como *IKEA Place*, *Pokémon GO* o plataformas de salud utilizan realidad aumentada o aprendizaje automático para transformar la experiencia del usuario. Establezca esta conexión con la vida real desde el inicio para generar interés y contexto.
2. Explique el propósito del tema.
   Aclárese que el objetivo consiste en comprender cómo se integran el *Machine Learning* (ML) y la Realidad Aumentada (AR) en iOS, y cómo herramientas como *Core ML* y *ARKit* permiten construir aplicaciones inteligentes, inmersivas y funcionales.
3. Introduzca los fundamentos del aprendizaje automático en iOS.
   Descríbanse las etapas del ciclo de vida de un modelo de ML: recopilación de datos, entrenamiento (con *Create ML*), exportación como *.mlmodel*, compilación con Xcode (*.mlmodelc*) e inferencia con *Core ML*. Destáquese el papel invisible pero esencial del subsistema *CO-ML* para ejecutar modelos de forma eficiente y privada en el dispositivo.
4. Profundice en el funcionamiento de *CO-ML*.
   Explíquese que *CO-ML* decide qué parte del hardware utilizar (CPU, GPU, ANE) para ejecutar modelos de ML. Resáltese como beneficios la velocidad, eficiencia energética y privacidad. Utilícense tablas comparativas para mostrar qué tareas se asignan a cada procesador.
5. Presente los fundamentos de la realidad aumentada con *ARKit*.
   Explíquese qué es la AR y cómo *ARKit* permite integrar contenido 3D en el entorno físico. Descríbanse las clases principales (*ARSession*, *ARAnchor*, entre otras), la detección de planos y el uso de *RealityKit* para renderizado y animaciones.
6. Demuestre diferencias clave entre *ARKit* y *QuickLook*.
   Aclárese que *QuickLook* permite visualizar archivos *USDZ* sin necesidad de escribir código, mientras que *ARKit* es un *framework* completo para crear experiencias interactivas. Utilícense ejemplos como “ver un mueble” frente a “jugar un videojuego interactivo con objetos en AR”.
7. Guíe la creación de una aplicación AR paso a paso.
   Muéstrese cómo configurar una aplicación básica con *ARKit*, incluyendo permisos de cámara, estructura de vistas e integración con SwiftUI usando *UIViewRepresentable*. Proporciónense ejemplos de código comentado para las vistas personalizadas de la 1 a la 6.

8. Destaque el uso de *Reality Composer*.
   Explíquese cómo crear modelos propios en formato *.usdz* con *Reality Composer*. Muéstrese cómo importar, animar y exportar para utilizarlos en *ARKit*. Invítese a los estudiantes a diseñar objetos educativos o de entretenimiento.

Tema 11 Accesibilidad y diseño inclusivo en aplicaciones móviles

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Inicie la sesión fomentando la empatía digital.
   Compártase una historia o ejemplo de una persona que no puede utilizar una aplicación común debido a una discapacidad visual o motriz. Establézcase que el objetivo no consiste únicamente en cumplir requisitos técnicos, sino en garantizar el acceso igualitario a la tecnología.
2. Explique los fundamentos de la accesibilidad en iOS.
   Preséntese el ecosistema de accesibilidad de Apple:

- VoiceOver
- Switch Control
- AssistiveTouch
- Zoom
- Voice Control
  Utilícese la tabla comparativa del material para mostrar sus usos. Muéstrese cómo implementar atributos como *accessibilityLabel*, *accessibilityHint* y *accessibilityTraits* en SwiftUI.

3. Enseñe las mejores prácticas para crear aplicaciones accesibles.
   Ilústrense las ocho prácticas fundamentales integradas en el ejemplo del material:
   - Lenguaje claro
   - Contraste adecuado
   - Alternativas al color
   - Imágenes con etiquetas
   - Dynamic Type
   - Retroalimentación accesible
   - Evitar gestos complejos
   - Orden lógico de navegación
     Realícense ejemplos en vivo con Xcode y el simulador para demostrar cómo escala el texto, cómo *VoiceOver* lee los elementos y cómo se responde con retroalimentación háptica o auditiva.
4. Aborde el diseño para discapacidades específicas.
   Organícese el contenido por tipo de discapacidad:
   - Visuales: uso de *VoiceOver*, etiquetas claras, contraste y *Dynamic Type*
   - Auditivas: subtítulos, retroalimentación visual, *Live Captions*
   - Motoras: evitar gestos complejos, uso de *AssistiveTouch*, botones grandes, tiempo de respuesta flexible

Invítese a reflexionar: ¿Qué barreras se eliminan cuando una aplicación presenta un diseño inclusivo adecuado?

5. Muestre las herramientas de prueba en Xcode.
   Demuéstrese cómo utilizar el *Accessibility Inspector* y cómo activar *VoiceOver* en el simulador.
   Guíese paso a paso:
   - Xcode > Open Developer Tool > Accessibility Inspector
   - Modo “Point to Inspect”
   - Validación de orden lógico, etiquetas, *hints* y *traits*

   Explíquese también el uso de UIAccessibility.post(notification:) y cómo integrar pruebas automatizadas de accesibilidad con *XCTest*.
6. Cierre con una reflexión guiada.
   Pregúntese:

- ¿Qué se aprendió hoy sobre las barreras tecnológicas invisibles?
- ¿Cómo se puede lograr que una próxima aplicación funcione para todas las personas?
- ¿Qué se sintió al usar *VoiceOver* o *Switch Control*?

Tema 12 Prácticas de accesibilidad en iOS

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Inicie la sesión contextualizando la accesibilidad como un imperativo ético, legal y de mercado.
   Explíquese que una de cada siete personas vive con alguna discapacidad y que, según la OMS y Google (2025), las aplicaciones que ignoran la accesibilidad pueden perder hasta el 40 % de su audiencia potencial en tres meses. Destáquese la relevancia de normativas como *WCAG 2.2*, *Material Design 3* y la *Ley General de Inclusión Digital*, y cómo la accesibilidad impacta la competitividad y la reputación de las aplicaciones.
2. Explique los fundamentos de accesibilidad y la compatibilidad con lectores de pantalla.
   Demuéstrese cómo *TalkBack* (Android) y *VoiceOver* (iOS) permiten a millones de personas con discapacidad visual interactuar con aplicaciones móviles. Ilústrese el uso de etiquetado semántico (*contentDescription* en XML, *semantics* en Compose), el manejo lógico del foco y la navegación por grupos de elementos relacionados. Realícense ejemplos prácticos de implementación y validación con herramientas como *Accessibility Scanner* y *Lint* de Android Studio.
3. Presente las mejores prácticas para crear aplicaciones accesibles.
   Abórdense principios clave como el contraste mínimo de 4.5:1, objetivos táctiles de al menos 48×48 dp, navegación predecible y soporte para múltiples modos de entrada. Realícense ejercicios prácticos de ajuste de contraste, etiquetado y validación automatizada. Resáltese la importancia de construir la accesibilidad desde el diseño, y no como un añadido posterior.
4. Analice el diseño inclusivo para personas con discapacidades visuales, auditivas y motoras.
   Explíquense técnicas como alternativas visuales y hápticas para alertas, subtítulos en contenido multimedia, soporte para *Switch Access* y navegación por voz. Realícense ejemplos de adaptación de

interfaz para distintos perfiles de usuario y promuévase la empatía técnica como base de una inclusión efectiva.

5. Demuestre el uso de herramientas de prueba de accesibilidad en Android Studio y dispositivos reales. Enséñese a utilizar *Lint*, *Accessibility Scanner*, el panel de accesibilidad en el *Layout Editor* y pruebas manuales con *TalkBack*. Explíquese cómo integrar pruebas automatizadas con *Espresso* y subráyese la importancia de validar con personas usuarias reales para identificar barreras no detectadas por herramientas automáticas.
6. Guíe la optimización de interfaces para diferentes tamaños y resoluciones de pantalla. Explíquese el uso de *WindowSizeClass* en *Jetpack Compose*, *ConstraintLayout*, recursos alternativos y simuladores de densidad para asegurar la adaptabilidad de la aplicación en móviles, tabletas y dispositivos plegables. Realícense ejercicios prácticos de diseño responsivo y adaptación visual.
7. Sugiera recursos de apoyo y fomente la reflexión crítica. Recomiéndense videos, lecturas y pódcast sobre accesibilidad, diseño inclusivo y tendencias legales y técnicas. Planteen preguntas como:

- ¿Qué cambios priorizarías al rediseñar una aplicación bancaria para personas con discapacidad visual?
- ¿Cómo equilibrarías la estética visual con los requisitos de contraste de *WCAG*?
- ¿Qué criterios utilizarías ante un conflicto entre innovación visual y normas de accesibilidad?

Tema 13 Gestión del ciclo de vida de aplicaciones móviles

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Inicie la sesión explicando la continuidad del desarrollo tras la publicación de una aplicación. Contextualícese que, una vez lanzada una aplicación, comienza una etapa fundamental: la gestión de su ciclo de vida. Explíquese cómo el mantenimiento, las actualizaciones, el control de versiones, la detección de errores y las pruebas permiten que la aplicación se mantenga útil, estable y segura con el paso del tiempo.
2. Explique el concepto de versionado semántico y los tipos de actualizaciones. Preséntese la estructura *MAJOR.MINOR.PATCH* y el significado de cada componente. Utilícense ejemplos concretos y clasifíquense los tipos de actualizaciones: incrementales, menores, mayores y de seguridad. Refuércese con tablas visuales que ilustren los escenarios.
3. Demuestre herramientas para la distribución de versiones. Explíquese el uso de *App Store Connect* y *TestFlight*. Indíquese cómo se configuran los metadatos, las fases de lanzamiento (*phased release*), y las pruebas beta internas y externas. Inclúyase una actividad con *TestFlight* para distribuir una versión beta entre los compañeros.
4. Presente herramientas de automatización (*CI/CD*) aplicables a iOS. Muéstrese el funcionamiento de *Xcode Cloud* para ejecutar pruebas y subir versiones al *App Store* de forma automatizada. Menciónense otras opciones como *Fastlane* y *Bitrise*. Realícese una explicación clara de cómo se integran con *GitHub*.
5. Explique el uso de Git para desarrolladores y diseñadores. Expóngase qué es *Git*, cómo se crean ramas, se realizan *commits* y se aplican *merges*. Utilícese la

analogía de una libreta y fotografías del progreso para facilitar la comprensión. Menciónense herramientas como *GitHub Desktop* y el control de versiones visual disponible en *Xcode*.

6. Incluya también el uso de Git para diseñadores.
   Explíquese el control de versiones de archivos en *Figma*, *Sketch* o *Adobe XD*, el uso de *Git LFS* y la sincronización con el equipo de desarrollo.
7. Demuestre el uso de pruebas unitarias y de integración en Xcode.
   Preséntense ejemplos con *XCTest*, *XCTAssert*, *setUp()* y *tearDown()*. Explíquese la diferencia entre pruebas unitarias (validación de funciones individuales) y de integración (interacción entre módulos). Ejecuten pruebas simples con *Command + U* o desde el menú de *Xcode*.
8. Enseñe el manejo de errores con Swift.
   Utilícense ejemplos reales con *throw*, *try*, *catch* y *do*. Explíquese cómo crear errores personalizados con *enum* y cómo gestionar condiciones anómalas con *try?* y *try!*
   Inclúyase una función como *iniciarSesion()* con errores comunes (usuario no encontrado, contraseña incorrecta) y demuéstrese su captura y la forma de comunicar el problema al usuario.

Tema 14 Despliegue y publicación de aplicaciones en App Store

Notas para la enseñanza del tema

Al profesor impartidor, se le recomienda lo siguiente:

1. Inicie la sesión conectando con el momento final del desarrollo.
   Explíquese que el proceso de publicación no es solo técnico, sino también estratégico. Utilícese una narrativa breve: “Tu aplicación está lista, pero ¿cómo llega a millones de usuarios?”. Destáquese que Apple exige calidad, seguridad y una experiencia adecuada para el usuario desde la etapa de publicación.
2. Explique el flujo completo del despliegue.
   Preséntense las etapas: registro en el *Apple Developer Program*, configuración en Xcode, pruebas en *TestFlight* y publicación en *App Store Connect*. Utilícese un gráfico *SmartArt* con este flujo para facilitar la comprensión. Refuércese la importancia del cumplimiento de normativas.
3. Demuestre cómo inscribirse en el *Apple Developer Program*.
   Muestren las diferencias entre una cuenta individual y una empresarial. Destáquense los requisitos administrativos y técnicos, como el *D-U-N-S Number* y el *Apple ID* con autenticación de doble factor. Explíquense los pasos desde el sitio web hasta la validación de la cuenta.
4. Presente la configuración técnica en Xcode.
   Enseñe qué es el *Bundle Identifier* y cómo generar un certificado de firma válido. Guíese al grupo paso a paso por la pestaña “*Signing & Capabilities*” y explíquense los errores más comunes. Demuéstrese cómo probar la aplicación en un dispositivo real antes de subirla.
5. Explique el uso de *TestFlight* para pruebas beta.
   Describa cómo distribuir versiones internas y externas de una aplicación con *TestFlight*. Resáltese la importancia de recibir retroalimentación real y registrar errores antes del lanzamiento oficial. Promuévase una actividad de simulación de pruebas con compañeros como personas testers.

6. Enseñe a crear metadatos y optimizar la visibilidad de la aplicación.
   Explíquese qué es la *App Store Optimization* (ASO). Detállese cómo redactar una descripción efectiva, elegir palabras clave relevantes y crear capturas de pantalla atractivas. Utilícense ejemplos como "EcoTrack" para mostrar cómo destacar las funciones principales de la aplicación.
7. Describa estrategias de mantenimiento posterior al lanzamiento.
   Abórdense los tipos de mantenimiento: correctivo, adaptativo, perfectivo y preventivo. Menciónense herramientas como *Firebase Crashlytics* para monitoreo y *Xcode Cloud* para automatización. Refuércese la necesidad de actualizaciones periódicas para fidelizar a las personas usuarias.
8. Propicie una reflexión guiada sobre la experiencia posterior a la publicación.
   Planteen preguntas como:

- ¿Qué desafíos enfrentas al publicar una aplicación?
- ¿Qué importancia tiene la retroalimentación del usuario?
- ¿Cómo puede un buen diseño de metadatos impactar el éxito de una aplicación?

Evidencia

Avance 1

Los estudiantes desarrollarán progresivamente un proyecto de aplicación móvil nativa para iOS, integrando los conocimientos adquiridos a lo largo del semestre. Este avance tiene como propósito guiar a los equipos en la planificación, diseño e implementación gradual de su aplicación, a partir de una problemática real vinculada con los Objetivos de Desarrollo Sostenible (ODS) propuestos por la ONU en su Agenda 2030.

Desde el inicio, se recomienda que el profesor fomente la reflexión sobre el papel de la tecnología en la transformación social, ambiental o económica. Los equipos (máximo cuatro integrantes) seleccionarán un ODS de su interés (como salud y bienestar, educación de calidad, igualdad de género o acción por el clima) e investigarán una necesidad concreta que pueda atenderse mediante una solución tecnológica.

A lo largo del avance, los estudiantes documentarán su proceso creativo, identificarán al usuario objetivo, definirán funcionalidades clave, diseñarán la interfaz en SwiftUI y establecerán la arquitectura básica de la aplicación.

El rol del docente será acompañar y retroalimentar cada etapa, promoviendo la colaboración, la investigación aplicada y el desarrollo de prototipos funcionales. Además, pueden programarse revisiones parciales o "entregas sprint" para asegurar el cumplimiento de cada fase.

Esta etapa concluirá con una versión preliminar de la aplicación, que servirá como base para el proyecto final.

Entrega final

El proyecto final consistirá en la entrega y presentación formal de una aplicación móvil nativa desarrollada en Swift, que proponga una solución concreta a un reto alineado con uno de los ODS de la Agenda 2030. Este producto final deberá consolidar lo aprendido durante todo el curso, tanto en el ámbito técnico como en aspectos de diseño ético, accesible e inclusivo.

Además de su entrega académica, los proyectos finales participarán en un concurso interno de aplicaciones, en el que se evaluarán todas las aplicaciones desarrolladas por los equipos del curso. Este concurso tiene como propósito promover la innovación, la motivación y el espíritu emprendedor entre los estudiantes. El jurado podrá estar integrado por profesores de la institución y profesionales invitados del área tecnológica o social.

Los criterios de evaluación incluirán:

- Alineación con el ODS y relevancia del problema abordado.
- Calidad técnica del desarrollo.
- Accesibilidad y diseño centrado en el usuario.
- Creatividad, innovación y viabilidad de la solución.
- Claridad y calidad de la presentación final.

Los equipos presentarán su aplicación funcional, explicarán su propósito y proceso de desarrollo, y responderán preguntas del jurado. Se otorgarán reconocimientos a las mejores aplicaciones en diversas categorías: impacto social, diseño, accesibilidad, solución más innovadora, entre otras.