Referencial de competencias profesionales

Formación Desarrollo de Aplicaciones Móviles

País: Spain

Profesiones cubiertas: Desarrollo de Aplicaciones Móviles

Este referencial es una certificación profesional privada, inspirada en los estándares de los referenciales de certificación franceses (metodología France Compétences). No constituye un referencial estatal.

Generado el 08/06/2026

Índice

ANEXO I Presentación sintética del marco del diploma

PRESENTACIÓN GENERAL

Nombre de la formación: Certificado de Profesionalidad en Desarrollo de Aplicaciones Móviles

Tipo de certificación: Certificado de Profesionalidad

Modalidades de formación:

Presencial, semipresencial y a distancia según modalidad de impartición

Finalidad y objetivos:

Capacitar profesionales para el análisis, diseño, desarrollo, prueba y publicación de aplicaciones móviles en plataformas Android e iOS, aplicando metodologías ágiles, patrones arquitectónicos modernos y principios de seguridad. Los desarrolladores serán capaces de traducir requisitos de negocio en soluciones técnicas escalables, implementar funcionalidades complejas, integrar servicios externos, realizar pruebas exhaustivas y publicar aplicaciones en tiendas oficiales.

Contexto profesional:

El desarrollo de aplicaciones móviles es un sector en crecimiento exponencial en España, impulsado por la transformación digital de empresas y la demanda de soluciones móviles innovadoras. Los desarrolladores de aplicaciones móviles son profesionales altamente demandados en empresas de tecnología, agencias digitales, startups y departamentos de TI de grandes organizaciones. Este certificado prepara profesionales para trabajar en equipos multidisciplinarios, colaborar con diseñadores y gestores de producto, y contribuir al desarrollo de aplicaciones que impactan millones de usuarios.

Público objetivo y condiciones de acceso:

Profesionales con conocimientos previos de programación que desean especializarse en desarrollo de aplicaciones móviles, así como desarrolladores web que buscan expandir sus competencias hacia plataformas móviles. También dirigido a técnicos en informática que desean adquirir competencias especializadas en desarrollo móvil.

Prerrequisitos:

- Conocimientos básicos de programación orientada a objetos
- Familiaridad con conceptos de desarrollo de software
- Comprensión de estructuras de datos y algoritmos básicos
- Experiencia previa en desarrollo de aplicaciones (web o escritorio recomendada)

Vías de acceso:

- formation_continue
- enseignement_a_distance

ORGANIZACIÓN DE LA FORMACIÓN

Enfoque pedagógico:

La formación combina metodología teórico-práctica con énfasis en aprendizaje experiencial mediante proyectos reales. Se utiliza enfoque de aprendizaje basado en problemas donde estudiantes desarrollan aplicaciones móviles completas desde requisitos hasta publicación.

- Aprendizaje centrado en competencias profesionales reales del sector

- Desarrollo de proyectos progresivos que aumentan en complejidad
- Integración de herramientas y tecnologías actuales del mercado
- Énfasis en buenas prácticas de desarrollo, seguridad y mantenibilidad
- Evaluación continua mediante proyectos prácticos y pruebas técnicas
- Desarrollo de habilidades blandas como comunicación y trabajo en equipo
- Preparación para certificaciones profesionales en plataformas móviles

Métodos de aprendizaje:

- Clases teóricas con exposición de conceptos y patrones de desarrollo
- Laboratorios prácticos con desarrollo de aplicaciones móviles
- Proyectos integradores que simulan escenarios reales de desarrollo
- Análisis de casos de estudio de aplicaciones móviles exitosas
- Trabajo colaborativo en equipos de desarrollo
- Revisión de código y feedback de pares
- Integración continua y automatización de pruebas
- Mentoría y seguimiento personalizado del progreso

Puntos fuertes de la formación:

- Formación integral que cubre todo el ciclo de vida del desarrollo móvil desde requisitos hasta publicación
- Énfasis en patrones arquitectónicos modernos y buenas prácticas de desarrollo
- Integración de seguridad como aspecto transversal en todas las fases
- Desarrollo de competencias en múltiples plataformas (Android, iOS, multiplataforma)
- Proyectos prácticos que simulan escenarios reales de desarrollo profesional
- Preparación para trabajar en equipos ágiles y metodologías de desarrollo iterativo
- Competencias en testing automatizado, depuración y optimización de rendimiento
- Capacidad de traducir requisitos de negocio en soluciones técnicas escalables

TABLA DE SÍNTESIS DE ACTIVIDADES - BLOQUES DE COMPETENCIAS - UNIDADES

Actividades	Bloques de competencias	Unidades
Función 1: Análisis y Especificación de Requisitos de Aplicaciones Móviles • Recopilar y analizar requisitos funcionales de la aplicación móvil • Identificar y documentar requisitos no funcionales de la aplicación • Definir alcance, priorizar requisitos y traducir necesidades de negocio	**Bloque 1 - Análisis y Especificación de Requisitos de Aplicaciones Móviles** • Recopilar y documentar requisitos funcionales de la aplicación móvil con stakeholders para asegurar la alineación con objetivos de negocio • Analizar y especificar requisitos no funcionales de rendimiento, seguridad y compatibilidad para definir restricciones técnicas móviles • Definir y negociar el alcance del producto móvil con stakeholders para establecer límites claros y gestionar cambios • Priorizar historias de usuario y funcionalidades según valor de negocio y esfuerzo técnico para optimizar entregas incrementales • Traducir necesidades de negocio en especificaciones técnicas funcionales para facilitar comunicación entre equipos de negocio y desarrollo	**Unidad U1 Análisis y Especificación de Requisitos de Aplicaciones Móviles**

Actividades	Bloques de competencias	Unidades
Función 2: Diseño de Arquitectura e Interfaz de Aplicaciones Móviles • Diseñar arquitectura de información y crear prototipos de pantallas • Validar wireframes y diseñar interfaces de usuario adaptativas • Integrar principios de experiencia de usuario y diseño centrado en el usuario • Crear componentes reutilizables y gestionar sistema de diseño	**Bloque 3 - Diseño de Arquitectura e Interfaz de Aplicaciones Móviles** • Diseñar arquitectura de información y modelado de datos para estructurar la aplicación móvil de forma escalable y mantenible • Implementar lógica de negocio aplicando patrones de arquitectura (MVC, MVP, MVVM) para asegurar separación de responsabilidades y mantenibilidad • Implementar navegación entre pantallas de la aplicación móvil gestionando flujos de usuario y paso de parámetros para crear experiencia coherente • Desarrollar interfaces de usuario móviles siguiendo directrices de diseño (Material Design, HIG) para asegurar consistencia y accesibilidad • Construir layouts adaptativos y responsivos para múltiples tamaños de pantalla utilizando unidades relativas y restricciones de layout • Crear componentes reutilizables de interfaz documentando API y manteniendo biblioteca centralizada para optimizar desarrollo y consistencia • Gestionar estado de la aplicación móvil implementando patrones de gestión (Redux, BLoC, Provider) para sincronizar datos entre componentes • Integrar patrones de diseño centrado en el usuario recopilando feedback y validando interacciones para iterar basándose en necesidades reales • Diseñar prototipos de pantallas móviles de baja y alta fidelidad utilizando herramientas de diseño para validar conceptos antes de desarrollo • Validar wireframes y prototipos con stakeholders obteniendo aprobación formal antes de iniciar desarrollo de funcionalidades	**Unidad U3 Diseño de Arquitectura e Interfaz de Aplicaciones Móviles**
Función 3: Selección de Plataforma y Configuración del Entorno de Desarrollo • Evaluar y seleccionar plataformas objetivo para el desarrollo • Seleccionar enfoque de desarrollo (nativo o multiplataforma) • Configurar entorno de desarrollo e instalar dependencias	**Bloque 2 - Selección de Plataforma y Configuración del Entorno de Desarrollo** • Evaluar y seleccionar plataforma de desarrollo (Android, iOS o multiplataforma) basándose en requisitos técnicos y restricciones de negocio • Configurar entorno de desarrollo móvil instalando IDEs, emuladores y dependencias para preparar infraestructura de codificación • Gestionar dependencias y SDK del proyecto móvil resolviendo conflictos de compatibilidad para mantener estabilidad de compilación	**Unidad U2 Selección de Plataforma y Configuración del Entorno de Desarrollo**

Actividades	Bloques de competencias	Unidades
Función 4: Desarrollo de Funcionalidades y Lógica de Negocio • Programar lógica de negocio e implementar navegación • Desarrollar interfaces de usuario y layouts adaptativos • Crear componentes reutilizables y gestionar estado de la aplicación • Implementar formularios, validaciones y animaciones	**Bloque 4 - Desarrollo de Funcionalidades y Lógica de Negocio** • Implementar persistencia local de datos utilizando bases de datos móviles (SQLite, Room, Core Data) con gestión de migraciones • Conectar aplicación móvil con APIs externas implementando clientes HTTP y gestionando errores de red para asegurar comunicación confiable • Consumir servicios web REST desde aplicación móvil implementando métodos HTTP y abstrayendo lógica de red en repositorios • Procesar respuestas JSON de servicios remotos deserializando datos y mapeándolos a modelos de aplicación con validación • Desarrollar formularios de entrada de datos con validación, gestión de teclado virtual y navegación accesible entre campos • Validar campos y entradas del usuario implementando validaciones cliente y servidor con mensajes de error contextuales	**Unidad U4 Desarrollo de Funcionalidades y Lógica de Negocio**
Función 5: Integración de Datos, Servicios y Funcionalidades Avanzadas • Implementar persistencia local y conectar con APIs externas • Consumir servicios REST y procesar respuestas JSON • Integrar autenticación, sesiones y notificaciones push • Integrar funcionalidades de hardware del dispositivo	**Bloque 5 - Integración de Servicios Avanzados y Funcionalidades Especializadas** • Integrar autenticación de usuarios implementando flujos OAuth, Firebase Auth o autenticación personalizada con recuperación de contraseña • Gestionar sesiones y tokens de acceso almacenando de forma segura y renovando automáticamente para mantener autenticación persistente • Implementar notificaciones push integrando Firebase Cloud Messaging o Apple Push Notification service con gestión de permisos • Añadir animaciones e interacciones en interfaz móvil utilizando APIs nativas optimizando rendimiento para mantener fluidez • Aplicar principios de seguridad en desarrollo identificando vulnerabilidades OWASP Mobile e implementando cifrado y almacenamiento seguro	**Unidad U5 Integración de Servicios Avanzados y Funcionalidades Especializadas**

Actividades	Bloques de competencias	Unidades
Función 6: Pruebas, Depuración y Optimización de Aplicaciones Móviles • Ejecutar pruebas funcionales y de compatibilidad • Escribir pruebas unitarias y automatizadas de interfaz • Depurar errores, gestionar excepciones y optimizar rendimiento • Monitorizar incidencias y corregir errores en producción	**Bloque 6 - Pruebas, Depuración y Optimización de Aplicaciones Móviles** • Gestionar errores y excepciones de ejecución implementando manejo robusto con logging y análisis para mejorar confiabilidad • Depurar fallos y errores de aplicación móvil utilizando herramientas de depuración, análisis de logs y reproducción de errores • Escribir pruebas unitarias para lógica de negocio utilizando frameworks de testing (JUnit, XCTest, Flutter Test) con cobertura adecuada • Desarrollar pruebas automatizadas de interfaz de usuario implementando escenarios de prueba con herramientas de testing (Espresso, XCUITest, Appium) • Ejecutar pruebas de compatibilidad en matriz de dispositivos documentando problemas y priorizando defectos para asegurar funcionamiento multiplataforma	**Unidad U6 Pruebas, Depuración y Optimización de Aplicaciones Móviles**
Función 7: Publicación, Actualización y Mantenimiento de Aplicaciones Móviles • Preparar compilaciones de producción y generar artefactos • Publicar aplicaciones en tiendas móviles • Actualizar aplicaciones y mantener compatibilidad con nuevas versiones	**Bloque 7 - Publicación, Actualización y Mantenimiento de Aplicaciones Móviles** • Preparar compilaciones para publicación configurando parámetros de producción, generando artefactos y gestionando claves de firma	**Unidad U7 Publicación, Actualización y Mantenimiento de Aplicaciones Móviles**
Función 8: Seguridad, Internacionalización y Gestión del Ciclo de Vida • Aplicar principios de seguridad e identificar vulnerabilidades • Gestionar ciclo de vida de la aplicación e implementar internacionalización • Añadir animaciones e interacciones avanzadas	**Bloque 8 - Seguridad, Internacionalización y Gestión del Ciclo de Vida**	**Unidad U8 Seguridad, Internacionalización y Gestión del Ciclo de Vida**
Función 9: Documentación, Control de Versiones e Integración Continua • Documentar arquitectura y participar en revisiones de código • Integrar control de versiones y gestionar branching • Configurar pipelines de integración y entrega continua	**Bloque 9 - Documentación, Control de Versiones e Integración Continua**	**Unidad U9 Documentación, Control de Versiones e Integración Continua**

Tabla de síntesis de funciones y actividades

Función 1: Análisis y Especificación de Requisitos de Aplicaciones Móviles
Actividad 1.1 Recopilar y analizar requisitos funcionales de la aplicación móvil
Actividad 1.2 Identificar y documentar requisitos no funcionales de la aplicación
Actividad 1.3 Definir alcance, priorizar requisitos y traducir necesidades de negocio
Función 2: Diseño de Arquitectura e Interfaz de Aplicaciones Móviles
Actividad 2.1 Diseñar arquitectura de información y crear prototipos de pantallas
Actividad 2.2 Validar wireframes y diseñar interfaces de usuario adaptativas
Actividad 2.3 Integrar principios de experiencia de usuario y diseño centrado en el usuario
Actividad 2.4 Crear componentes reutilizables y gestionar sistema de diseño
Función 3: Selección de Plataforma y Configuración del Entorno de Desarrollo
Actividad 3.1 Evaluar y seleccionar plataformas objetivo para el desarrollo
Actividad 3.2 Seleccionar enfoque de desarrollo (nativo o multiplataforma)
Actividad 3.3 Configurar entorno de desarrollo e instalar dependencias
Función 4: Desarrollo de Funcionalidades y Lógica de Negocio
Actividad 4.1 Programar lógica de negocio e implementar navegación
Actividad 4.2 Desarrollar interfaces de usuario y layouts adaptativos
Actividad 4.3 Crear componentes reutilizables y gestionar estado de la aplicación
Actividad 4.4 Implementar formularios, validaciones y animaciones
Función 5: Integración de Datos, Servicios y Funcionalidades Avanzadas
Actividad 5.1 Implementar persistencia local y conectar con APIs externas
Actividad 5.2 Consumir servicios REST y procesar respuestas JSON
Actividad 5.3 Integrar autenticación, sesiones y notificaciones push
Actividad 5.4 Integrar funcionalidades de hardware del dispositivo
Función 6: Pruebas, Depuración y Optimización de Aplicaciones Móviles
Actividad 6.1 Ejecutar pruebas funcionales y de compatibilidad
Actividad 6.2 Escribir pruebas unitarias y automatizadas de interfaz
Actividad 6.3 Depurar errores, gestionar excepciones y optimizar rendimiento
Actividad 6.4 Monitorizar incidencias y corregir errores en producción
Función 7: Publicación, Actualización y Mantenimiento de Aplicaciones Móviles
Actividad 7.1 Preparar compilaciones de producción y generar artefactos
Actividad 7.2 Publicar aplicaciones en tiendas móviles
Actividad 7.3 Actualizar aplicaciones y mantener compatibilidad con nuevas versiones
Función 8: Seguridad, Internacionalización y Gestión del Ciclo de Vida

Actividad 8.1 Aplicar principios de seguridad e identificar vulnerabilidades
Actividad 8.2 Gestionar ciclo de vida de la aplicación e implementar internacionalización
Actividad 8.3 Añadir animaciones e interacciones avanzadas
Función 9: Documentación, Control de Versiones e Integración Continua
Actividad 9.1 Documentar arquitectura y participar en revisiones de código
Actividad 9.2 Integrar control de versiones y gestionar branching
Actividad 9.3 Configurar pipelines de integración y entrega continua

Estas áreas, desglosadas en actividades que el profesional realiza con autonomía o bajo supervisión, pueden no ejercerse todas en la entidad empleadora, especialmente en un primer empleo.

ANEXO II Marco de actividades profesionales

Contexto profesional y empleos

El desarrollo de aplicaciones móviles es una disciplina especializada que abarca el análisis, diseño, implementación, prueba y publicación de aplicaciones de software para dispositivos móviles. Los desarrolladores de aplicaciones móviles trabajan en la creación de soluciones que funcionan en plataformas como Android e iOS, considerando restricciones técnicas específicas de dispositivos móviles como capacidad de procesamiento limitada, memoria reducida, consumo de batería y conectividad variable. Este rol requiere comprensión profunda de ecosistemas móviles, patrones arquitectónicos modernos, metodologías ágiles y principios de seguridad. Los desarrolladores colaboran estrechamente con diseñadores de interfaz, gestores de producto, especialistas en QA y otros miembros del equipo para entregar aplicaciones de alta calidad que satisfacen necesidades de usuarios y objetivos de negocio.

La persona titulada desempeña diferentes funciones:
– análisis y especificación de requisitos de aplicaciones móviles ;
– diseño de arquitectura e interfaz de aplicaciones móviles ;
– selección de plataforma y configuración del entorno de desarrollo ;
– desarrollo de funcionalidades y lógica de negocio ;
– integración de datos, servicios y funcionalidades avanzadas ;
– pruebas, depuración y optimización de aplicaciones móviles ;
– publicación, actualización y mantenimiento de aplicaciones móviles ;
– seguridad, internacionalización y gestión del ciclo de vida ;
– documentación, control de versiones e integración continua ;

Empleos relacionados

Los empleos de estos profesionales se encuentran en distintas estructuras públicas y privadas, entre ellas:
– Empresas de desarrollo de software especializadas en aplicaciones móviles ;
– Agencias digitales y de publicidad con capacidades de desarrollo móvil ;
– Departamentos de TI y desarrollo de grandes empresas ;
– Startups tecnológicas enfocadas en soluciones móviles ;
– Consultorías de transformación digital ;
– Equipos internos de desarrollo en empresas de diversos sectores ;
– Trabajo autónomo como desarrollador freelance ;

Los puestos reciben denominaciones diferentes según el sector. Algunos ejemplos son:
– Desarrollador de Aplicaciones Móviles ;
– Ingeniero de Software Móvil ;
– Especialista en Desarrollo Android ;
– Especialista en Desarrollo iOS ;
– Desarrollador Multiplataforma ;
– Arquitecto de Aplicaciones Móviles ;
– Líder Técnico de Desarrollo Móvil ;
– Desarrollador Full Stack Móvil ;

Compensación de Entrada

Los desarrolladores de aplicaciones móviles junior en España tienen salarios iniciales que oscilan entre 22.000 y 28.000 euros anuales, dependiendo de ubicación geográfica, tamaño de empresa y experiencia previa. Desarrolladores con experiencia previa en programación pueden acceder a posiciones con compensación inicial más alta. Empresas en grandes ciudades tecnológicas (Madrid, Barcelona) ofrecen compensación superior a la media nacional.

Evolución profesional (3 a 5 años)

Los desarrolladores de aplicaciones móviles pueden evolucionar hacia roles de mayor responsabilidad y especialización. La trayectoria profesional típica comienza con desarrollador junior, progresa hacia desarrollador senior, y puede derivar hacia liderazgo técnico, arquitectura de software o gestión de equipos. Alternativamente, pueden especializarse en áreas como seguridad móvil, rendimiento, o arquitectura de sistemas distribuidos.

- Desarrollador Senior de Aplicaciones Móviles con responsabilidad en diseño arquitectónico
- Arquitecto de Aplicaciones Móviles definiendo estándares técnicos y patrones
- Líder Técnico de equipo de desarrollo móvil
- Especialista en Seguridad de Aplicaciones Móviles
- Especialista en Rendimiento y Optimización de Aplicaciones Móviles
- Gestor de Producto Técnico con enfoque en desarrollo móvil
- Director de Ingeniería de Aplicaciones Móviles

Continuación de estudios

Los desarrolladores pueden continuar su formación mediante certificaciones profesionales en plataformas específicas (Google Associate Android Developer, Apple Certified Associate Developer), cursos especializados en seguridad móvil, arquitectura de microservicios, o programas de máster en ingeniería de software. También pueden explorar especializaciones en áreas emergentes como desarrollo de aplicaciones para wearables, realidad aumentada o inteligencia artificial en dispositivos móviles.

ÉTICA Y DEONTOLOGÍA PROFESIONAL

El desarrollo de aplicaciones móviles conlleva responsabilidades éticas significativas relacionadas con privacidad de usuarios, seguridad de datos y impacto social de las aplicaciones.

- Protección de privacidad de usuarios mediante implementación de medidas de seguridad robustas y cumplimiento de regulaciones como RGPD
- Transparencia en recopilación y uso de datos personales, informando claramente a usuarios sobre qué datos se recopilan y cómo se utilizan
- Seguridad de datos mediante cifrado, almacenamiento seguro y protección contra vulnerabilidades conocidas
- Accesibilidad inclusiva asegurando que aplicaciones son usables por personas con diferentes capacidades y discapacidades
- Responsabilidad social considerando impacto de aplicaciones en sociedad y evitando contenido discriminatorio o dañino
- Cumplimiento normativo adhiriéndose a regulaciones de protección de datos, accesibilidad y requisitos de tiendas de aplicaciones
- Integridad profesional manteniendo estándares de calidad, documentación clara y comunicación honesta sobre capacidades y limitaciones

Función 1: Análisis y Especificación de Requisitos de Aplicaciones Móviles

Función 1: Análisis y Especificación de Requisitos de Aplicaciones Móviles

Actividad 1.1 Recopilar y analizar requisitos funcionales de la aplicación móvil

- Realizar sesiones de elicitación de requisitos con stakeholders y usuarios finales
  - ■ Conducir entrevistas estructuradas con product owners y usuarios clave
  - ■ Facilitar talleres de requisitos con equipos de negocio y técnicos
  - ■ Documentar necesidades expresadas y casos de uso identificados
- Documentar funcionalidades esperadas en forma de casos de uso e historias de usuario
  - ■ Redactar casos de uso con actores, precondiciones y flujos de interacción
  - ■ Crear historias de usuario con criterios de aceptación medibles
  - ■ Definir escenarios de éxito y casos de error para cada funcionalidad
- Validar la comprensión de requisitos con el equipo de producto y desarrollo
  - ■ Presentar requisitos documentados para revisión y aprobación
  - ■ Resolver ambigüedades y conflictos de interpretación
  - ■ Obtener consenso sobre el alcance funcional de cada iteración
- Organizar requisitos en un backlog estructurado y priorizado
  - ■ Agrupar funcionalidades relacionadas en épicas y features
  - ■ Establecer dependencias entre requisitos
  - ■ Crear trazabilidad entre requisitos y objetivos de negocio

■ Medios y recursos

- Herramientas de gestión de requisitos (Jira, Azure DevOps, Trello)
- Plantillas de casos de uso y historias de usuario
- Documentación de procesos de negocio existentes
- Acceso a stakeholders y usuarios finales para entrevistas
- Herramientas de colaboración (Confluence, Miro, Mural)

■ Resultados esperados

- Documento de requisitos funcionales completo y validado
- Backlog de historias de usuario con criterios de aceptación claros
- Matriz de trazabilidad entre requisitos y objetivos de negocio
- Aprobación formal de requisitos por parte de stakeholders

Función 1: Análisis y Especificación de Requisitos de Aplicaciones Móviles

Actividad 1.2 Identificar y documentar requisitos no funcionales de la aplicación

- Identificar requisitos no funcionales como rendimiento, seguridad, escalabilidad y usabilidad
  - ■ Definir métricas de rendimiento (tiempo de carga, FPS, consumo de memoria)
  - ■ Especificar requisitos de seguridad y cumplimiento normativo
  - ■ Establecer objetivos de escalabilidad y capacidad de usuarios concurrentes
- Evaluar restricciones técnicas del entorno móvil objetivo
  - ■ Analizar limitaciones de hardware (memoria, procesador, batería)
  - ■ Considerar restricciones de conectividad y ancho de banda
  - ■ Evaluar limitaciones de sistemas operativos soportados

- Asegurar que los requisitos no funcionales sean medibles y verificables
  - ■ Definir criterios de aceptación cuantitativos para cada requisito
  - ■ Establecer métodos de medición y herramientas de validación
  - ■ Crear casos de prueba para verificar cumplimiento de requisitos
- Documentar requisitos no funcionales en especificación técnica
  - ■ Redactar especificación de requisitos no funcionales
  - ■ Incluir restricciones técnicas y limitaciones identificadas
  - ■ Validar especificación con equipo técnico y stakeholders

■ Medios y recursos

- Herramientas de análisis de rendimiento (Android Profiler, Instruments)
- Documentación de estándares de seguridad móvil (OWASP Mobile Top 10)
- Especificaciones técnicas de plataformas Android e iOS
- Herramientas de monitorización (Firebase, Sentry)
- Plantillas de requisitos no funcionales

■ Resultados esperados

- Documento de requisitos no funcionales completo y cuantificado
- Métricas de rendimiento, seguridad y escalabilidad definidas
- Criterios de aceptación medibles para cada requisito no funcional
- Plan de validación de requisitos no funcionales

Función 1: Análisis y Especificación de Requisitos de Aplicaciones Móviles

Actividad 1.3 Definir alcance, priorizar requisitos y traducir necesidades de negocio

- Delimitar funcionalidades incluidas y excluidas en cada iteración
  - ■ Definir MVP (Producto Mínimo Viable) con funcionalidades críticas
  - ■ Identificar funcionalidades opcionales y de futuro
  - ■ Documentar exclusiones explícitas para evitar desviaciones
- Negociar alcance con equipo de negocio y responsables del proyecto
  - ■ Presentar propuestas de alcance con análisis de esfuerzo
  - ■ Resolver conflictos entre requisitos y restricciones
  - ■ Obtener acuerdo formal sobre alcance de cada fase
- Priorizar historias de usuario según valor de negocio y esfuerzo técnico
  - ■ Aplicar técnicas de priorización (MoSCoW, Value vs Effort)
  - ■ Evaluar dependencias técnicas entre historias
  - ■ Revisar y ajustar priorización en cada ciclo de desarrollo
- Traducir objetivos de negocio en especificaciones técnicas funcionales
  - ■ Interpretar necesidades de negocio en términos técnicos
  - ■ Facilitar comunicación entre área de negocio y desarrollo
  - ■ Asegurar trazabilidad entre objetivos de negocio y funcionalidades

■ Medios y recursos

- Herramientas de gestión de proyectos (Jira, Azure DevOps)
- Matriz de priorización (MoSCoW, Value vs Effort)
- Documentación de objetivos de negocio
- Acceso a product owners y stakeholders
- Plantillas de especificación técnica

■ Resultados esperados

• Documento de alcance acordado y aprobado
• Backlog priorizado según valor de negocio y esfuerzo
• Especificaciones técnicas de funcionalidades de negocio
• Plan de entregas por fases con hitos definidos

Función 2: Diseño de Arquitectura e Interfaz de Aplicaciones Móviles

Función 2: Diseño de Arquitectura e Interfaz de Aplicaciones Móviles

Actividad 2.1 Diseñar arquitectura de información y crear prototipos de pantallas

- Estructurar la organización de contenidos y flujos de navegación de la aplicación
  - ■ Crear mapas de sitio (sitemaps) con jerarquía de pantallas
  - ■ Definir flujos de navegación principales y alternativos
  - ■ Documentar transiciones entre pantallas y estados de la interfaz
- Definir modelos de datos y relaciones entre entidades
  - ■ Crear diagramas entidad-relación (ER) de la arquitectura de datos
  - ■ Especificar atributos y tipos de datos de cada entidad
  - ■ Documentar relaciones y restricciones de integridad
- Crear prototipos de baja y alta fidelidad de las pantallas
  - ■ Diseñar wireframes de baja fidelidad para validar flujos
  - ■ Crear prototipos de alta fidelidad con diseño visual detallado
  - ■ Representar estados de interfaz (carga, error, vacío, éxito)
- Documentar la arquitectura para guiar el desarrollo del equipo
  - ■ Redactar especificación de arquitectura de información
  - ■ Incluir diagramas de flujo de navegación
  - ■ Crear guía de componentes y patrones de interfaz

■ Medios y recursos

- Herramientas de diseño (Figma, Sketch, Adobe XD)
- Herramientas de prototipado (Axure, InVision, Framer)
- Plantillas de wireframes y prototipos
- Guías de diseño de Material Design y Human Interface Guidelines
- Herramientas de diagramación (Lucidchart, Draw.io)

■ Resultados esperados

- Mapa de sitio (sitemap) con jerarquía de pantallas definida
- Prototipos de baja fidelidad validados con stakeholders
- Prototipos de alta fidelidad con diseño visual completo
- Documentación de arquitectura de información y flujos de navegación

Función 2: Diseño de Arquitectura e Interfaz de Aplicaciones Móviles

Actividad 2.2 Validar wireframes y diseñar interfaces de usuario adaptativas

- Presentar wireframes al equipo de desarrollo y clientes para recoger feedback
  - ■ Organizar sesiones de revisión de wireframes con stakeholders
  - ■ Documentar observaciones y cambios solicitados
  - ■ Iterar sobre wireframes incorporando feedback
- Obtener aprobación formal de wireframes como base para el desarrollo
  - ■ Consolidar cambios acordados en versión final
  - ■ Obtener firma o aprobación explícita de stakeholders
  - ■ Comunicar wireframes aprobados al equipo de desarrollo

- Diseñar layouts responsivos que se adapten a diferentes resoluciones
  - ■ Crear layouts para múltiples tamaños de pantalla (móvil, tablet)
  - ■ Utilizar unidades relativas y sistemas de cuadrícula
  - ■ Definir puntos de quiebre (breakpoints) para adaptabilidad
- Asegurar coherencia visual y accesibilidad en toda la interfaz
  - ■ Definir paleta de colores y tipografía consistente
  - ■ Establecer criterios de contraste y legibilidad
  - ■ Documentar pautas de accesibilidad (WCAG, ATAG)

■ Medios y recursos

- Herramientas de diseño colaborativo (Figma, Miro)
- Guías de accesibilidad (WCAG 2.1, ATAG)
- Herramientas de prueba de accesibilidad (Axe, WAVE)
- Especificaciones de Material Design y Human Interface Guidelines
- Plantillas de sistemas de diseño

■ Resultados esperados

- Wireframes aprobados formalmente por stakeholders
- Especificación de layouts adaptativos para múltiples dispositivos
- Guía de accesibilidad implementada en diseños
- Documentación de cambios y decisiones de diseño

Función 2: Diseño de Arquitectura e Interfaz de Aplicaciones Móviles

Actividad 2.3 Integrar principios de experiencia de usuario y diseño centrado en el usuario

- Aplicar metodologías de diseño centrado en el usuario (UCD) durante el desarrollo
  - ■ Realizar investigación de usuarios (entrevistas, encuestas, observación)
  - ■ Crear personas de usuario basadas en datos reales
  - ■ Definir escenarios de uso y casos de uso críticos
- Incorporar feedback de usuarios reales en decisiones de diseño
  - ■ Conducir pruebas de usabilidad con usuarios reales
  - ■ Recopilar feedback mediante encuestas y análisis de comportamiento
  - ■ Iterar sobre diseños basándose en feedback recibido
- Implementar principios de usabilidad en la interfaz
  - ■ Asegurar consistencia en patrones de interacción
  - ■ Proporcionar retroalimentación clara al usuario
  - ■ Prevenir errores mediante validaciones y confirmaciones
- Validar que patrones de interacción responden a necesidades reales
  - ■ Realizar pruebas de usabilidad iterativas
  - ■ Medir métricas de usabilidad (tasa de error, tiempo de tarea)
  - ■ Documentar lecciones aprendidas para futuras iteraciones

■ Medios y recursos

- Herramientas de investigación de usuarios (UserTesting, Maze, Validately)
- Herramientas de análisis de comportamiento (Hotjar, Mixpanel)
- Plantillas de personas de usuario
- Guías de principios de usabilidad (Nielsen, Shneiderman)
- Herramientas de prototipado interactivo (Figma, Framer)

■ Resultados esperados

- Personas de usuario documentadas basadas en investigación
- Escenarios de uso y casos de uso críticos definidos
- Resultados de pruebas de usabilidad documentados
- Iteraciones de diseño basadas en feedback de usuarios

Función 2: Diseño de Arquitectura e Interfaz de Aplicaciones Móviles

Actividad 2.4 Crear componentes reutilizables y gestionar sistema de diseño

- Identificar elementos de UI repetidos y abstraerlos en componentes
  - ■ Analizar diseños para identificar patrones repetidos
  - ■ Definir componentes base (botones, campos, tarjetas, etc.)
  - ■ Especificar variantes y estados de cada componente
- Documentar la API de cada componente para facilitar su uso
  - ■ Crear especificación de propiedades y parámetros
  - ■ Documentar ejemplos de uso y casos de aplicación
  - ■ Definir restricciones y limitaciones de cada componente
- Mantener una biblioteca de componentes coherente con el sistema de diseño
  - ■ Crear y mantener biblioteca de componentes en herramienta de diseño
  - ■ Versionar componentes y documentar cambios
  - ■ Comunicar actualizaciones al equipo de desarrollo
- Asegurar consistencia visual en toda la aplicación
  - ■ Definir tokens de diseño (colores, tipografía, espaciado)
  - ■ Crear guía de estilos visual
  - ■ Revisar diseños para asegurar adherencia a sistema de diseño

■ Medios y recursos

- Herramientas de diseño con soporte de componentes (Figma, Sketch)
- Herramientas de documentación de componentes (Storybook, Zeroheight)
- Especificaciones de Material Design y Human Interface Guidelines
- Herramientas de control de versiones para diseño (Abstract, Figma versioning)
- Plantillas de documentación de componentes

■ Resultados esperados

- Biblioteca de componentes reutilizables documentada
- Especificación de API de componentes clara y accesible
- Guía de estilos visual con tokens de diseño definidos
- Sistema de diseño mantenido y versionado

Función 3: Selección de Plataforma y Configuración del Entorno de Desarrollo

Función 3: Selección de Plataforma y Configuración del Entorno de Desarrollo

Actividad 3.1 Evaluar y seleccionar plataformas objetivo para el desarrollo

- Evaluar características del ecosistema Android y su adecuación al proyecto
  - ■ Analizar cuota de mercado y distribución de versiones de Android
  - ■ Evaluar características de hardware y capacidades del ecosistema
  - ■ Considerar restricciones de Google Play y políticas de distribución
- Evaluar características del ecosistema iOS y su adecuación al proyecto
  - ■ Analizar cuota de mercado y versiones mínimas de iOS soportadas
  - ■ Revisar directrices de Apple y requisitos de App Store
  - ■ Evaluar restricciones de dispositivos Apple (iPhone, iPad)
- Analizar restricciones técnicas del entorno móvil objetivo
  - ■ Evaluar limitaciones de hardware (memoria, procesador, batería)
  - ■ Considerar restricciones de conectividad y ancho de banda
  - ■ Analizar implicaciones de versiones mínimas de SO soportadas
- Documentar decisión de plataforma y sus implicaciones técnicas
  - ■ Redactar justificación de selección de plataforma(s)
  - ■ Documentar versiones mínimas y máximas de SO soportadas
  - ■ Especificar implicaciones técnicas y restricciones identificadas

■ Medios y recursos

- Datos de cuota de mercado (Statista, IDC, Counterpoint)
- Documentación oficial de Android y iOS
- Directrices de Google Play y App Store
- Herramientas de análisis de mercado móvil
- Especificaciones técnicas de dispositivos

■ Resultados esperados

- Análisis comparativo de ecosistemas Android e iOS
- Decisión documentada de plataforma(s) objetivo
- Especificación de versiones mínimas y máximas de SO
- Documento de implicaciones técnicas y restricciones

Función 3: Selección de Plataforma y Configuración del Entorno de Desarrollo

Actividad 3.2 Seleccionar enfoque de desarrollo (nativo o multiplataforma)

- Evaluar ventajas del desarrollo nativo en términos de rendimiento y UX
  - ■ Analizar rendimiento de aplicaciones nativas vs multiplataforma
  - ■ Evaluar acceso a APIs del sistema y funcionalidades de hardware
  - ■ Considerar experiencia de usuario nativa en cada plataforma
- Evaluar frameworks multiplataforma según requisitos del proyecto
  - ■ Analizar opciones (Flutter, React Native, Xamarin, Kotlin Multiplatform)
  - ■ Evaluar impacto en rendimiento, mantenibilidad y cobertura
  - ■ Considerar curva de aprendizaje y disponibilidad de talento

- Comparar enfoque nativo frente a alternativas multiplataforma
  - ■ Crear matriz de comparación (rendimiento, coste, tiempo, mantenimiento)
  - ■ Analizar casos de uso específicos del proyecto
  - ■ Evaluar riesgos y beneficios de cada enfoque
- Justificar y documentar la elección tecnológica ante el equipo
  - ■ Redactar documento de decisión arquitectónica (ADR)
  - ■ Presentar análisis y justificación al equipo técnico
  - ■ Obtener consenso sobre enfoque seleccionado

■ Medios y recursos

- Documentación de frameworks multiplataforma (Flutter, React Native, Xamarin)
- Herramientas de benchmarking de rendimiento
- Análisis de casos de estudio de aplicaciones publicadas
- Plantilla de Decisión Arquitectónica (ADR)
- Acceso a expertos en diferentes tecnologías

■ Resultados esperados

- Análisis comparativo de enfoques nativo vs multiplataforma
- Documento de Decisión Arquitectónica (ADR) justificado
- Especificación de framework y lenguajes de programación seleccionados
- Plan de capacitación para el equipo en tecnología seleccionada

Función 3: Selección de Plataforma y Configuración del Entorno de Desarrollo

Actividad 3.3 Configurar entorno de desarrollo e instalar dependencias

- Instalar y configurar el IDE adecuado según la plataforma objetivo
  - ■ Instalar Android Studio para desarrollo Android
  - ■ Instalar Xcode para desarrollo iOS
  - ■ Configurar extensiones y plugins necesarios en el IDE
- Establecer emuladores, dispositivos virtuales y conexiones físicas
  - ■ Configurar emuladores de Android con diferentes versiones de SO
  - ■ Configurar simuladores de iOS con diferentes modelos de dispositivo
  - ■ Establecer conexión con dispositivos físicos para pruebas
- Identificar e instalar librerías, SDKs y herramientas necesarias
  - ■ Instalar SDKs de plataforma (Android SDK, iOS SDK)
  - ■ Instalar gestores de paquetes (Gradle, CocoaPods, npm, pub)
  - ■ Instalar herramientas de desarrollo (Git, emuladores, debuggers)
- Verificar que el entorno funciona correctamente
  - ■ Ejecutar proyecto de prueba en emulador y dispositivo físico
  - ■ Validar que todas las herramientas están correctamente configuradas
  - ■ Documentar pasos de configuración para nuevos miembros del equipo

■ Medios y recursos

- Android Studio y Xcode
- SDKs de plataforma (Android SDK, iOS SDK)
- Gestores de paquetes (Gradle, CocoaPods, npm, pub)
- Emuladores y simuladores
- Documentación oficial de plataformas
- Herramientas de control de versiones (Git)

■ **Resultados esperados**

- IDE instalado y configurado correctamente
- Emuladores y simuladores funcionales
- Dispositivos físicos conectados y funcionales
- Proyecto de prueba ejecutándose exitosamente
- Documentación de configuración del entorno

Función 4: Desarrollo de Funcionalidades y Lógica de Negocio

Función 4: Desarrollo de Funcionalidades y Lógica de Negocio

Actividad 4.1 Programar lógica de negocio e implementar navegación

- Implementar las reglas de negocio y flujos de trabajo en el lenguaje seleccionado
  - ■ Codificar lógica de negocio según especificaciones de requisitos
  - ■ Implementar validaciones y reglas de negocio
  - ■ Crear modelos de datos que representen entidades de negocio
- Estructurar el código siguiendo patrones de arquitectura (MVC, MVP, MVVM)
  - ■ Separar responsabilidades entre capas (presentación, lógica, datos)
  - ■ Implementar patrón arquitectónico seleccionado
  - ■ Crear interfaces y abstracciones para facilitar testing
- Definir y codificar flujos de navegación entre vistas
  - ■ Implementar componentes de navegación nativos o del framework
  - ■ Gestionar historial de navegación y back stack
  - ■ Pasar parámetros entre pantallas de forma segura
- Garantizar separación de responsabilidades entre capas
  - ■ Implementar inyección de dependencias
  - ■ Crear servicios y repositorios reutilizables
  - ■ Documentar arquitectura de capas implementada

■ Medios y recursos

- IDE de desarrollo (Android Studio, Xcode, VS Code)
- Lenguajes de programación (Kotlin, Swift, Dart, JavaScript)
- Frameworks de arquitectura (Clean Architecture, MVVM, BLoC)
- Herramientas de análisis de código (SonarQube, Lint)
- Documentación de patrones de diseño

■ Resultados esperados

- Lógica de negocio implementada según especificaciones
- Código estructurado siguiendo patrón arquitectónico
- Navegación entre pantallas funcional
- Separación clara de responsabilidades entre capas
- Código revisado y aprobado en code review

Función 4: Desarrollo de Funcionalidades y Lógica de Negocio

Actividad 4.2 Desarrollar interfaces de usuario y layouts adaptativos

- Construir pantallas respetando guías de diseño de cada plataforma
  - ■ Implementar Material Design para Android
  - ■ Implementar Human Interface Guidelines para iOS
  - ■ Utilizar componentes nativos de cada plataforma
- Implementar componentes visuales definidos en prototipos
  - ■ Crear layouts XML o SwiftUI según plataforma
  - ■ Implementar componentes de formularios, listas y tarjetas
  - ■ Aplicar estilos y temas definidos en sistema de diseño
- Diseñar e implementar layouts responsivos para múltiples tamaños

- Utilizar unidades relativas (dp, sp) en lugar de píxeles
- Implementar restricciones de layout (ConstraintLayout, AutoLayout)
- Probar interfaz en dispositivos de distintos tamaños y orientaciones

• Asegurar coherencia visual y accesibilidad en toda la interfaz
- Aplicar paleta de colores y tipografía consistente
- Implementar contraste adecuado para legibilidad
- Añadir etiquetas de accesibilidad (contentDescription, VoiceOver)

■ Medios y recursos

• IDE de desarrollo (Android Studio, Xcode)
• Herramientas de diseño (Figma, Sketch)
• Guías de Material Design y Human Interface Guidelines
• Librerías de componentes (Material Components, SwiftUI)
• Herramientas de prueba de accesibilidad

■ Resultados esperados

• Interfaces de usuario implementadas según prototipos
• Layouts responsivos que se adaptan a múltiples tamaños
• Componentes visuales coherentes con sistema de diseño
• Interfaz accesible según estándares WCAG
• Pruebas de interfaz en múltiples dispositivos exitosas

Función 4: Desarrollo de Funcionalidades y Lógica de Negocio

Actividad 4.3 Crear componentes reutilizables y gestionar estado de la aplicación

• Identificar elementos de UI repetidos y abstraerlos en componentes
- Analizar código para identificar patrones repetidos
- Crear componentes parametrizables y reutilizables
- Documentar API de cada componente

• Implementar solución de gestión de estado adecuada
- Seleccionar patrón de gestión de estado (Redux, BLoC, Provider, ViewModel)
- Implementar flujos de actualización de estado
- Gestionar estado compartido entre pantallas

• Garantizar coherencia del estado entre pantallas y componentes
- Centralizar estado en store o ViewModel
- Implementar listeners para cambios de estado
- Sincronizar estado entre componentes dependientes

• Depurar y optimizar flujos de actualización de estado
- Utilizar herramientas de debugging de estado (Redux DevTools, BLoC Inspector)
- Identificar actualizaciones innecesarias de estado
- Optimizar rendimiento de flujos de estado

■ Medios y recursos

• Librerías de gestión de estado (Redux, BLoC, Provider, GetX)
• Herramientas de debugging (Redux DevTools, BLoC Inspector)
• IDE de desarrollo con soporte de debugging
• Documentación de patrones de gestión de estado
• Herramientas de profiling de rendimiento

■ Resultados esperados

- Componentes reutilizables creados y documentados
- Sistema de gestión de estado implementado
- Estado coherente entre pantallas y componentes
- Flujos de estado optimizados y sin actualizaciones innecesarias
- Documentación de arquitectura de estado

Función 4: Desarrollo de Funcionalidades y Lógica de Negocio

Actividad 4.4 Implementar formularios, validaciones y animaciones

- Implementar formularios con campos, tipos de entrada y controles necesarios
  - ■ Crear layouts de formularios con campos de texto, selección, etc.
  - ■ Implementar tipos de entrada específicos (email, teléfono, fecha)
  - ■ Gestionar foco y navegación entre campos
- Implementar validaciones en el lado cliente para campos de formularios
  - ■ Validar formato de datos (email, teléfono, URL)
  - ■ Validar longitud y obligatoriedad de campos
  - ■ Mostrar mensajes de error claros y contextuales
- Implementar animaciones de transición y micro-interacciones
  - ■ Crear animaciones de transición entre pantallas
  - ■ Implementar micro-interacciones en componentes
  - ■ Utilizar APIs de animación nativas del framework
- Optimizar animaciones para evitar impacto en rendimiento
  - ■ Utilizar hardware acceleration cuando sea posible
  - ■ Reducir complejidad de animaciones en dispositivos de bajo rendimiento
  - ■ Medir y optimizar FPS de animaciones

■ Medios y recursos

- IDE de desarrollo (Android Studio, Xcode)
- Librerías de validación (Formz, Vuelidate)
- APIs de animación nativas (Android Animation, Core Animation)
- Herramientas de profiling (Android Profiler, Instruments)
- Documentación de patrones de formularios

■ Resultados esperados

- Formularios implementados con campos y tipos de entrada correctos
- Validaciones cliente funcionales con mensajes de error claros
- Animaciones e interacciones implementadas
- Rendimiento de animaciones optimizado (60 FPS)
- Experiencia de usuario fluida en formularios

Función 5: Integración de Datos, Servicios y Funcionalidades Avanzadas

Función 5: Integración de Datos, Servicios y Funcionalidades Avanzadas

Actividad 5.1 Implementar persistencia local y conectar con APIs externas

- Seleccionar e integrar solución de almacenamiento local
  - ■ Evaluar opciones (SQLite, Room, Core Data, Hive)
  - ■ Integrar librería de almacenamiento seleccionada
  - ■ Configurar base de datos local
- Diseñar esquema de datos local y operaciones CRUD
  - ■ Crear tablas y relaciones en base de datos local
  - ■ Implementar operaciones CRUD (Create, Read, Update, Delete)
  - ■ Crear DAOs o repositorios para acceso a datos
- Gestionar migración de datos entre versiones de la aplicación
  - ■ Implementar versionado de esquema de base de datos
  - ■ Crear scripts de migración para cambios de esquema
  - ■ Probar migraciones en diferentes escenarios
- Configurar clientes HTTP para comunicación con servicios externos
  - ■ Integrar librería HTTP (Retrofit, Alamofire, Dio)
  - ■ Configurar interceptores para autenticación y logging
  - ■ Implementar timeout y reintentos de conexión

■ Medios y recursos

- Librerías de almacenamiento (SQLite, Room, Core Data, Hive)
- Librerías HTTP (Retrofit, Alamofire, Dio, http)
- Herramientas de inspección de base de datos (DB Browser, Realm Studio)
- Herramientas de testing de APIs (Postman, Insomnia)
- Documentación de APIs externas

■ Resultados esperados

- Base de datos local implementada y funcional
- Operaciones CRUD implementadas y probadas
- Sistema de migración de datos implementado
- Clientes HTTP configurados y funcionales
- Documentación de esquema de datos y APIs

Función 5: Integración de Datos, Servicios y Funcionalidades Avanzadas

Actividad 5.2 Consumir servicios REST y procesar respuestas JSON

- Implementar capa de acceso a datos que consume servicios REST
  - ■ Crear repositorios que encapsulan llamadas a APIs
  - ■ Implementar métodos HTTP (GET, POST, PUT, DELETE)
  - ■ Gestionar códigos de respuesta HTTP
- Deserializar y mapear respuestas JSON a modelos de datos
  - ■ Crear modelos de datos que representen respuestas JSON
  - ■ Implementar serialización/deserialización JSON

    - Mapear campos JSON a propiedades de modelos
- Gestionar casos de respuestas incompletas o con formato inesperado
    - Implementar validación de respuestas JSON
    - Manejar respuestas nulas o incompletas
    - Implementar fallbacks para datos faltantes
- Implementar validaciones sobre datos recibidos
    - Validar tipos de datos recibidos
    - Validar rangos y restricciones de datos
    - Registrar errores de validación para debugging

■ Medios y recursos

- Librerías de serialización JSON (Gson, Moshi, Codable, json_serializable)
- Librerías HTTP (Retrofit, Alamofire, Dio)
- Herramientas de testing de APIs (Postman, Insomnia)
- Documentación de APIs REST
- Herramientas de análisis de respuestas JSON

■ Resultados esperados

- Repositorios implementados para acceso a APIs
- Modelos de datos creados para respuestas JSON
- Serialización/deserialización JSON funcional
- Validación de respuestas implementada
- Manejo de errores y casos excepcionales

Función 5: Integración de Datos, Servicios y Funcionalidades Avanzadas

Actividad 5.3 Integrar autenticación, sesiones y notificaciones push

- Implementar flujos de registro, inicio de sesión y recuperación de contraseña
    - Crear pantallas de registro y login
    - Implementar validación de credenciales
    - Crear flujo de recuperación de contraseña
- Integrar proveedores de autenticación externos
    - Integrar OAuth (Google, Facebook, Apple)
    - Integrar Firebase Authentication
    - Integrar LDAP o autenticación corporativa
- Implementar almacenamiento seguro de tokens de autenticación
    - Almacenar tokens en almacenamiento seguro del dispositivo
    - Implementar gestión de JWT
    - Renovar automáticamente tokens expirados
- Integrar servicios de notificaciones push
    - Integrar Firebase Cloud Messaging (FCM) para Android
    - Integrar Apple Push Notification service (APNs) para iOS
    - Gestionar permisos de notificaciones
    - Implementar manejo de notificaciones en primer y segundo plano

■ Medios y recursos

- Librerías de autenticación (Firebase Auth, OAuth libraries)
- Servicios de notificaciones (FCM, APNs)
- Almacenamiento seguro (Keychain, Keystore)
- Herramientas de testing de autenticación

• Documentación de proveedores de autenticación

■ **Resultados esperados**

• Flujos de autenticación implementados y funcionales
• Proveedores de autenticación externa integrados
• Tokens almacenados de forma segura
• Notificaciones push funcionales en ambas plataformas
• Gestión de sesiones y cierre de sesión por expiración

Función 5: Integración de Datos, Servicios y Funcionalidades Avanzadas

Actividad 5.4 Integrar funcionalidades de hardware del dispositivo

• Acceder y utilizar capacidades de hardware del dispositivo
  ■ Implementar acceso a cámara del dispositivo
  ■ Implementar acceso a GPS y localización
  ■ Implementar acceso a acelerómetro y sensores
  ■ Implementar acceso a micrófono y audio
• Gestionar permisos de acceso al hardware de forma correcta
  ■ Solicitar permisos de forma transparente al usuario
  ■ Implementar flujos de solicitud de permisos en tiempo de ejecución
  ■ Documentar permisos requeridos en manifest/Info.plist
• Implementar fallbacks para dispositivos sin hardware requerido
  ■ Detectar disponibilidad de hardware
  ■ Proporcionar alternativas cuando hardware no está disponible
  ■ Mostrar mensajes informativos al usuario
• Optimizar uso de hardware para evitar impacto en batería
  ■ Utilizar hardware de forma eficiente
  ■ Liberar recursos cuando no se usan
  ■ Implementar estrategias de ahorro de energía

■ **Medios y recursos**

• APIs de hardware (Camera API, Location API, Sensor API)
• Librerías de acceso a hardware (image_picker, geolocator, sensors_plus)
• Documentación de permisos de plataforma
• Herramientas de testing de hardware
• Emuladores con soporte de sensores

■ **Resultados esperados**

• Acceso a hardware del dispositivo implementado
• Permisos solicitados y gestionados correctamente
• Fallbacks implementados para hardware no disponible
• Uso de hardware optimizado para batería
• Documentación de funcionalidades de hardware

Función 6: Pruebas, Depuración y Optimización de Aplicaciones Móviles

Función 6: Pruebas, Depuración y Optimización de Aplicaciones Móviles

Actividad 6.1 Ejecutar pruebas funcionales y de compatibilidad

- Ejecutar pruebas funcionales específicas para el ecosistema Android
  - ■ Crear casos de prueba para funcionalidades críticas
  - ■ Ejecutar pruebas en diferentes versiones de Android
  - ■ Validar integración con servicios Android (Google Play Services, permisos)
- Ejecutar pruebas funcionales específicas para el ecosistema iOS
  - ■ Crear casos de prueba para funcionalidades críticas
  - ■ Ejecutar pruebas en diferentes versiones de iOS
  - ■ Validar integración con servicios Apple (Sign in with Apple, APNs)
- Planificar y ejecutar pruebas de compatibilidad en matriz de dispositivos
  - ■ Definir matriz de dispositivos (tamaños, versiones, fabricantes)
  - ■ Ejecutar pruebas en múltiples dispositivos y emuladores
  - ■ Documentar problemas de compatibilidad encontrados
- Priorizar y corregir problemas según impacto en experiencia de usuario
  - ■ Clasificar defectos por severidad
  - ■ Priorizar correcciones según impacto
  - ■ Verificar correcciones en múltiples dispositivos

■ Medios y recursos

- Herramientas de testing (Espresso, XCUITest, Appium)
- Emuladores y simuladores
- Dispositivos físicos de prueba
- Herramientas de gestión de casos de prueba (TestRail, Zephyr)
- Herramientas de reporte de defectos (Jira, Azure DevOps)

■ Resultados esperados

- Casos de prueba funcionales documentados
- Pruebas ejecutadas en múltiples dispositivos y versiones
- Problemas de compatibilidad identificados y documentados
- Defectos priorizados y asignados para corrección
- Reporte de pruebas con cobertura de funcionalidades

Función 6: Pruebas, Depuración y Optimización de Aplicaciones Móviles

Actividad 6.2 Escribir pruebas unitarias y automatizadas de interfaz

- Diseñar e implementar pruebas unitarias para lógica de aplicación
  - ■ Crear pruebas unitarias con frameworks (JUnit, XCTest, Flutter Test)
  - ■ Verificar comportamiento de unidades de código aisladas
  - ■ Implementar mocks y stubs para dependencias
- Implementar pruebas automatizadas de interfaz de usuario
  - ■ Crear pruebas de UI con Espresso (Android)
  - ■ Crear pruebas de UI con XCUITest (iOS)

    - Crear pruebas de UI con Appium (multiplataforma)
- Definir escenarios de prueba críticos para automatización
    - Identificar flujos de usuario críticos
    - Crear casos de prueba para escenarios principales
    - Documentar pasos de prueba y resultados esperados
- Integrar pruebas automatizadas en pipeline de CI/CD
    - Configurar ejecución automática de pruebas
    - Generar reportes de cobertura de pruebas
    - Establecer criterios de calidad para pasar pipeline

■ Medios y recursos

- Frameworks de testing (JUnit, XCTest, Flutter Test, Espresso, XCUITest)
- Herramientas de CI/CD (GitHub Actions, Bitrise, Jenkins)
- Herramientas de cobertura de código (Jacoco, Codecov)
- Emuladores y simuladores
- Documentación de frameworks de testing

■ Resultados esperados

- Pruebas unitarias implementadas con cobertura adecuada
- Pruebas automatizadas de UI funcionales
- Escenarios críticos cubiertos por automatización
- Pipeline CI/CD ejecutando pruebas automáticamente
- Reportes de cobertura y resultados de pruebas

Función 6: Pruebas, Depuración y Optimización de Aplicaciones Móviles

Actividad 6.3 Depurar errores, gestionar excepciones y optimizar rendimiento

- Utilizar herramientas de depuración para identificar y corregir errores
    - Utilizar debugger del IDE para inspeccionar código
    - Analizar trazas de error y stack traces
    - Reproducir y documentar pasos para reproducir errores
- Implementar sistema de manejo de errores y excepciones
    - Capturar y gestionar excepciones de forma controlada
    - Mostrar mensajes de error comprensibles al usuario
    - Registrar errores para análisis posterior
- Identificar cuellos de botella de rendimiento mediante profiling
    - Utilizar Android Profiler para análisis de rendimiento
    - Utilizar Instruments para análisis en iOS
    - Identificar métodos y operaciones lentas
- Optimizar tiempo de carga, fluidez de interfaz y uso de memoria
    - Optimizar tiempo de carga de pantallas
    - Mejorar fluidez de animaciones (60 FPS)
    - Reducir consumo de memoria y fugas de memoria

■ Medios y recursos

- Herramientas de depuración (IDE debugger, Logcat, Console)
- Herramientas de profiling (Android Profiler, Instruments, DevTools)
- Herramientas de análisis de memoria (Memory Profiler, Xcode Memory Graph)
- Herramientas de logging (Timber, os.log)
- Documentación de optimización de rendimiento

■ **Resultados esperados**

- Errores identificados y corregidos
- Sistema de manejo de excepciones implementado
- Cuellos de botella de rendimiento identificados
- Optimizaciones implementadas y verificadas
- Métricas de rendimiento mejoradas

Función 6: Pruebas, Depuración y Optimización de Aplicaciones Móviles

Actividad 6.4 Monitorizar incidencias y corregir errores en producción

- Integrar herramientas de monitorización y analítica en la aplicación
  - ■ Integrar Firebase Crashlytics para reporte de crashes
  - ■ Integrar Google Analytics para analítica de uso
  - ■ Integrar Sentry para monitorización de errores
- Revisar periódicamente dashboards de métricas y detectar anomalías
  - ■ Monitorizar tasa de crashes y errores
  - ■ Analizar métricas de uso y comportamiento de usuarios
  - ■ Identificar tendencias y anomalías en datos
- Analizar informes de errores reportados por usuarios
  - ■ Revisar informes de crashes y errores
  - ■ Reproducir fallos identificados
  - ■ Diagnosticar causa raíz de problemas
- Publicar actualizaciones correctivas y comunicar correcciones
  - ■ Corregir fallos identificados
  - ■ Publicar actualizaciones en tiendas
  - ■ Comunicar correcciones a usuarios afectados

■ **Medios y recursos**

- Herramientas de monitorización (Firebase Crashlytics, Sentry, Datadog)
- Herramientas de analítica (Google Analytics, Mixpanel, Amplitude)
- Herramientas de gestión de incidencias (Jira, Azure DevOps)
- Herramientas de comunicación (email, push notifications)
- Documentación de procesos de respuesta a incidencias

■ **Resultados esperados**

- Herramientas de monitorización integradas
- Dashboards de métricas configurados
- Alertas configuradas para anomalías
- Errores identificados y corregidos rápidamente
- Comunicación de correcciones a usuarios

Función 7: Publicación, Actualización y Mantenimiento de Aplicaciones Móviles

Función 7: Publicación, Actualización y Mantenimiento de Aplicaciones Móviles

Actividad 7.1 Preparar compilaciones de producción y generar artefactos

- Configurar variantes de compilación de producción
  - Crear configuraciones de build para producción
  - Configurar claves de firma y certificados
  - Establecer parámetros de compilación optimizados
- Generar artefactos de distribución listos para publicación
  - Generar APK y AAB para Android
  - Generar IPA para iOS
  - Verificar integridad de artefactos generados
- Verificar que compilación de producción supera todas las pruebas
  - Ejecutar suite de pruebas en compilación de producción
  - Validar que no hay errores de compilación
  - Verificar que todas las funcionalidades funcionan correctamente
- Documentar proceso de compilación y parámetros utilizados
  - Documentar pasos de compilación
  - Registrar versión de SDK y dependencias utilizadas
  - Crear checklist de verificación pre-publicación

Medios y recursos

- IDE de desarrollo (Android Studio, Xcode)
- Herramientas de compilación (Gradle, Xcode Build System)
- Herramientas de firma (jarsigner, codesign)
- Herramientas de CI/CD (GitHub Actions, Bitrise, Fastlane)
- Documentación de proceso de compilación

Resultados esperados

- Configuración de compilación de producción establecida
- Artefactos de distribución generados (APK, AAB, IPA)
- Compilación de producción verificada y probada
- Documentación de proceso de compilación
- Checklist de verificación pre-publicación completado

Función 7: Publicación, Actualización y Mantenimiento de Aplicaciones Móviles

Actividad 7.2 Publicar aplicaciones en tiendas móviles

- Preparar metadatos de la aplicación para publicación
  - Redactar descripción de aplicación
  - Crear capturas de pantalla y videos promocionales
  - Diseñar iconos y gráficos de portada
  - Seleccionar categoría y palabras clave
- Gestionar proceso de revisión y aprobación de tiendas
  - Enviar aplicación a Google Play para revisión

- Enviar aplicación a App Store para revisión
- Responder a comentarios de revisores
- Resolver problemas identificados en revisión

• Configurar distribución por fases y grupos de prueba
- Configurar lanzamiento gradual en Google Play
- Crear grupos de prueba (TestFlight para iOS)
- Monitorizar métricas durante lanzamiento gradual

• Completar lanzamiento completo en tiendas
- Expandir distribución a 100% de usuarios
- Monitorizar métricas post-lanzamiento
- Responder a comentarios y reseñas de usuarios

■ Medios y recursos

• Google Play Console
• Apple App Store Connect
• Herramientas de gestión de metadatos
• Herramientas de analítica (Google Analytics, App Store Analytics)
• Documentación de políticas de tiendas

■ Resultados esperados

• Metadatos de aplicación preparados y completos
• Aplicación aprobada en Google Play y App Store
• Lanzamiento gradual configurado y monitorizado
• Aplicación disponible para todos los usuarios
• Métricas de lanzamiento analizadas

Función 7: Publicación, Actualización y Mantenimiento de Aplicaciones Móviles

Actividad 7.3 Actualizar aplicaciones y mantener compatibilidad con nuevas versiones

• Monitorizar cambios en nuevas versiones de Android e iOS
- Revisar notas de lanzamiento de nuevas versiones de SO
- Identificar cambios que afecten a la aplicación
- Evaluar impacto de cambios en funcionalidades existentes

• Adaptar código y dependencias para mantener compatibilidad
- Actualizar código para cumplir con nuevos requisitos
- Actualizar dependencias a versiones compatibles
- Probar aplicación en nuevas versiones de SO

• Publicar actualizaciones antes de que versiones antiguas queden obsoletas
- Planificar actualizaciones según ciclo de lanzamiento de SO
- Publicar actualizaciones en tiendas
- Comunicar actualizaciones a usuarios

• Gestionar compatibilidad hacia atrás con versiones antiguas
- Mantener soporte para versiones mínimas de SO
- Implementar fallbacks para características no disponibles
- Documentar versiones de SO soportadas

■ Medios y recursos

• Documentación oficial de Android e iOS
• Herramientas de testing en múltiples versiones
• Emuladores y simuladores con diferentes versiones

- Herramientas de CI/CD para testing automático
- Herramientas de gestión de dependencias

■ **Resultados esperados**

- Cambios de nuevas versiones de SO identificados
- Código adaptado para nuevas versiones
- Dependencias actualizadas y compatibles
- Actualizaciones publicadas en tiendas
- Compatibilidad hacia atrás mantenida

Función 8: Seguridad, Internacionalización y Gestión del Ciclo de Vida

Función 8: Seguridad, Internacionalización y Gestión del Ciclo de Vida

Actividad 8.1 Aplicar principios de seguridad e identificar vulnerabilidades

- Identificar vulnerabilidades de seguridad comunes en aplicaciones móviles
  - Revisar OWASP Mobile Top 10
  - Realizar análisis de seguridad estático del código
  - Ejecutar pruebas de seguridad dinámicas
- Implementar buenas prácticas de cifrado y almacenamiento seguro
  - Implementar cifrado de datos en tránsito (HTTPS/TLS)
  - Implementar cifrado de datos en reposo
  - Utilizar almacenamiento seguro para datos sensibles (Keychain, Keystore)
- Implementar comunicación segura con servidores
  - Validar certificados SSL/TLS
  - Implementar certificate pinning
  - Utilizar protocolos seguros de comunicación
- Realizar revisiones de seguridad del código antes de publicación
  - Revisar código para vulnerabilidades de seguridad
  - Ejecutar herramientas de análisis de seguridad
  - Documentar hallazgos de seguridad y correcciones

Medios y recursos

- Documentación OWASP Mobile Top 10
- Herramientas de análisis de seguridad estático (SonarQube, Checkmarx)
- Herramientas de análisis de seguridad dinámico (Burp Suite, OWASP ZAP)
- Librerías de cifrado (Bouncy Castle, Tink, CryptoKit)
- Herramientas de testing de seguridad

Resultados esperados

- Vulnerabilidades de seguridad identificadas
- Cifrado implementado para datos sensibles
- Comunicación segura con servidores
- Revisiones de seguridad documentadas
- Vulnerabilidades corregidas antes de publicación

Función 8: Seguridad, Internacionalización y Gestión del Ciclo de Vida

Actividad 8.2 Gestionar ciclo de vida de la aplicación e implementar internacionalización

- Implementar gestión correcta del ciclo de vida de la aplicación
  - Implementar callbacks de ciclo de vida en Android (onCreate, onPause, onResume, onDestroy)
  - Implementar callbacks de ciclo de vida en iOS (viewDidLoad, viewWillAppear, viewDidDisappear)
  - Gestionar transiciones entre estados de la aplicación
- Asegurar que la aplicación guarda y restaura su estado
  - Implementar guardado de estado (savedInstanceState, NSCoding)
  - Restaurar estado cuando la aplicación se reanuda

    - Gestionar datos persistentes entre sesiones
- Gestionar procesos en segundo plano respetando restricciones de plataforma
    - Implementar servicios en segundo plano en Android
    - Implementar background tasks en iOS
    - Respetar restricciones de batería y recursos
- Preparar la aplicación para soportar múltiples idiomas y regiones
    - Externalizar cadenas de texto en archivos de recursos
    - Implementar detección automática de idioma del dispositivo
    - Permitir selección manual de idioma por usuario

■ Medios y recursos

- Documentación de ciclo de vida de Android e iOS
- Herramientas de localización (Lokalise, Crowdin, Phrase)
- Archivos de recursos de strings (strings.xml, Localizable.strings)
- Herramientas de testing de ciclo de vida
- Documentación de restricciones de background

■ Resultados esperados

- Ciclo de vida de aplicación implementado correctamente
- Estado de aplicación guardado y restaurado
- Procesos en segundo plano implementados
- Aplicación soporta múltiples idiomas
- Localización de recursos implementada

Función 8: Seguridad, Internacionalización y Gestión del Ciclo de Vida

Actividad 8.3 Añadir animaciones e interacciones avanzadas

- Implementar animaciones de transición entre pantallas
    - Crear animaciones de entrada y salida de pantallas
    - Implementar transiciones compartidas entre pantallas
    - Utilizar APIs de animación nativas del framework
- Implementar micro-interacciones en componentes
    - Crear animaciones de feedback en botones
    - Implementar animaciones de carga y progreso
    - Añadir efectos visuales en interacciones del usuario
- Utilizar APIs de animación nativas para garantizar fluidez
    - Utilizar Android Animation Framework
    - Utilizar Core Animation en iOS
    - Utilizar APIs de animación del framework (Flutter, React Native)
- Optimizar animaciones para evitar impacto en rendimiento
    - Utilizar hardware acceleration cuando sea posible
    - Reducir complejidad de animaciones en dispositivos de bajo rendimiento
    - Medir y optimizar FPS de animaciones

■ Medios y recursos

- APIs de animación (Android Animation, Core Animation, Flutter Animation)
- Herramientas de profiling (Android Profiler, Instruments)
- Librerías de animación (Lottie, Flare)
- Herramientas de diseño de animaciones (Principle, Framer)
- Documentación de APIs de animación

■ **Resultados esperados**

- Animaciones de transición implementadas
- Micro-interacciones añadidas a componentes
- Animaciones fluidas a 60 FPS
- Rendimiento optimizado sin impacto en batería
- Experiencia de usuario mejorada con animaciones

Función 9: Documentación, Control de Versiones e Integración Continua

Función 9: Documentación, Control de Versiones e Integración Continua

Actividad 9.1 Documentar arquitectura y participar en revisiones de código

- Redactar documentación técnica de la arquitectura de la aplicación
  - Documentar estructura de capas y componentes
  - Crear diagramas de arquitectura
  - Documentar patrones de diseño utilizados
- Documentar decisiones de diseño y justificación técnica
  - Crear Documentos de Decisión Arquitectónica (ADR)
  - Documentar alternativas consideradas
  - Justificar decisiones técnicas tomadas
- Mantener documentación actualizada a lo largo del proyecto
  - Actualizar documentación con cambios de arquitectura
  - Revisar y validar documentación periódicamente
  - Comunicar cambios al equipo
- Participar activamente en revisiones de código
  - Revisar código de otros miembros del equipo
  - Proporcionar feedback constructivo
  - Verificar cumplimiento de estándares de calidad

Medios y recursos

- Herramientas de documentación (Confluence, Notion, GitBook)
- Herramientas de diagramación (Lucidchart, Draw.io, Miro)
- Plataformas de control de versiones (GitHub, GitLab, Bitbucket)
- Herramientas de revisión de código (GitHub Review, GitLab MR)
- Plantillas de Documentación Arquitectónica

Resultados esperados

- Documentación técnica completa y actualizada
- Diagramas de arquitectura claros
- Documentos de Decisión Arquitectónica (ADR) documentados
- Documentación accesible para nuevos miembros del equipo
- Feedback constructivo en revisiones de código

Función 9: Documentación, Control de Versiones e Integración Continua

Actividad 9.2 Integrar control de versiones y gestionar branching

- Utilizar sistemas de control de versiones para gestionar código fuente
  - Configurar repositorio Git
  - Realizar commits con mensajes descriptivos
  - Mantener historial de cambios limpio y organizado
- Aplicar estrategias de branching adecuadas al equipo
  - Implementar GitFlow para proyectos complejos
  - Implementar trunk-based development para equipos ágiles

- Crear ramas para features, bugfixes y releases
- Gestionar merges y resolución de conflictos
  - Realizar merges de forma segura
  - Resolver conflictos de merge de forma correcta
  - Utilizar pull requests para revisión de código
- Gestionar etiquetado de versiones
  - Crear tags para versiones de lanzamiento
  - Documentar cambios en cada versión (CHANGELOG)
  - Mantener versionado semántico

Medios y recursos

- Sistemas de control de versiones (Git)
- Plataformas de hosting (GitHub, GitLab, Bitbucket)
- Herramientas de gestión de ramas (GitFlow, GitHub Flow)
- Herramientas de merge (Git, GitHub, GitLab)
- Documentación de estrategias de branching

Resultados esperados

- Repositorio Git configurado y funcional
- Estrategia de branching implementada
- Commits con mensajes descriptivos
- Merges realizados sin conflictos
- Versiones etiquetadas y documentadas

Función 9: Documentación, Control de Versiones e Integración Continua

Actividad 9.3 Configurar pipelines de integración y entrega continua

- Configurar herramientas de CI/CD para automatizar compilación y pruebas
  - Configurar GitHub Actions para CI/CD
  - Configurar Bitrise para compilación y distribución
  - Configurar Fastlane para automatización de tareas
- Definir stages del pipeline y criterios de calidad
  - Crear stage de compilación
  - Crear stage de pruebas unitarias
  - Crear stage de pruebas de integración
  - Crear stage de análisis de código
- Establecer criterios de calidad que deben superarse
  - Definir cobertura mínima de pruebas
  - Establecer reglas de análisis de código
  - Configurar alertas para fallos de calidad
- Mantener pipeline actualizado y funcional
  - Monitorizar ejecución del pipeline
  - Actualizar configuración según cambios del proyecto
  - Documentar pasos del pipeline

Medios y recursos

- Herramientas de CI/CD (GitHub Actions, Bitrise, Jenkins, GitLab CI)
- Herramientas de automatización (Fastlane, Gradle)
- Herramientas de análisis de código (SonarQube, Lint)
- Herramientas de cobertura (Jacoco, Codecov)

- Documentación de CI/CD

■ Resultados esperados

- Pipeline de CI/CD configurado y funcional
- Compilación automatizada en cada commit
- Pruebas ejecutadas automáticamente
- Análisis de código realizado automáticamente
- Reportes de calidad generados
- Distribución automatizada a tiendas

ANEXO III Marco de competencias

Bloque 1 - Análisis y Especificación de Requisitos de Aplicaciones Móviles

Objetivo pedagógico

Capacitar al desarrollador para recopilar, analizar y especificar requisitos funcionales y no funcionales de aplicaciones móviles, asegurando alineación con objetivos de negocio y restricciones técnicas.

Actividades

Actividad 1.1 Recopilar y analizar requisitos funcionales de la aplicación móvil
Actividad 1.2 Identificar y documentar requisitos no funcionales de la aplicación
Actividad 1.3 Definir alcance, priorizar requisitos y traducir necesidades de negocio

Competencias

Competencias	Indicadores
C1 - Recopilar y documentar requisitos funcionales de la aplicación móvil con stakeholders para asegurar la alineación con objetivos de negocio	Realiza entrevistas estructuradas con al menos 3 stakeholders diferentes documentando requisitos en formato de historias de usuario Elabora matriz de trazabilidad que vincula cada requisito funcional con objetivos de negocio y criterios de aceptación medibles
C2 - Analizar y especificar requisitos no funcionales de rendimiento, seguridad y compatibilidad para definir restricciones técnicas móviles	Identifica y documenta al menos 5 requisitos no funcionales (rendimiento, seguridad, escalabilidad) con métricas cuantificables Elabora especificación técnica que incluye criterios de aceptación para cada requisito no funcional y restricciones de dispositivos móviles
C3 - Definir y negociar el alcance del producto móvil con stakeholders para establecer límites claros y gestionar cambios	Documenta alcance inicial con lista de funcionalidades incluidas y excluidas, obteniendo aprobación formal de al menos 2 stakeholders Implementa proceso de gestión de cambios que registra todas las solicitudes de cambio de alcance con impacto estimado
C4 - Priorizar historias de usuario y funcionalidades según valor de negocio y esfuerzo técnico para optimizar entregas incrementales	Aplica matriz de priorización (valor vs. esfuerzo) a un mínimo de 20 historias de usuario, documentando justificación de cada prioridad Gestiona dependencias técnicas entre historias y comunica roadmap priorizado a equipo de desarrollo con estimaciones de esfuerzo
C5 - Traducir necesidades de negocio en especificaciones técnicas funcionales para facilitar comunicación entre equipos de negocio y desarrollo	Convierte al menos 10 requisitos de negocio en especificaciones técnicas detalladas con casos de uso y flujos de datos Facilita sesiones de alineación entre stakeholders de negocio y equipo técnico, documentando acuerdos y decisiones

Saberes asociados

Recopilación y Documentación de Requisitos Funcionales

Técnicas de recopilación de requisitos

- Entrevistas con stakeholders
- Talleres de requisitos
- Análisis de documentación existente
- Observación de usuarios
- Cuestionarios y encuestas

Documentación de casos de uso

- Estructura de casos de uso
- Flujos normales y excepcionales
- Actores y precondiciones
- Postcondiciones y resultados

Redacción de historias de usuario

- Formato de historias de usuario
- Criterios de aceptación
- Estimación de complejidad
- Dependencias entre historias

Validación de requisitos

- Verificación de completitud
- Validación con stakeholders
- Identificación de ambigüedades
- Confirmación de comprensión

Análisis de Requisitos No Funcionales

Requisitos de rendimiento

- Tiempo de respuesta
- Consumo de memoria
- Duración de batería
- Velocidad de carga
- Métricas de rendimiento

Requisitos de seguridad

- Autenticación y autorización
- Protección de datos sensibles
- Cumplimiento normativo
- Vulnerabilidades OWASP Mobile
- Cifrado de datos

Requisitos de compatibilidad

- Versiones de sistema operativo
- Tamaños de pantalla
- Capacidades de dispositivos
- Resoluciones de pantalla
- Orientación de pantalla

Restricciones técnicas móviles

- Limitaciones de procesador
- Limitaciones de memoria RAM
- Limitaciones de almacenamiento
- Conectividad de red
- Consumo de datos

Definición y Negociación de Alcance

Definición de alcance del producto

- Identificación de funcionalidades incluidas
- Identificación de funcionalidades excluidas
- Límites del proyecto
- Entregables principales
- Fases del proyecto

Negociación de requisitos

- Identificación de conflictos
- Presentación de opciones
- Evaluación de impacto
- Búsqueda de consenso
- Documentación de acuerdos

Gestión de cambios de alcance

- Proceso de solicitud de cambios
- Evaluación de impacto en cronograma
- Evaluación de impacto en recursos
- Aprobación de cambios
- Comunicación de cambios

Documentación de alcance

- Especificación de alcance
- Matriz de trazabilidad
- Documento de requisitos
- Aprobación de stakeholders
- Actualización de documentación

Priorización de Historias de Usuario y Funcionalidades

Priorización de backlog

- Criterios de priorización
- Matriz de priorización
- Ordenamiento de backlog
- Revisión de prioridades
- Comunicación de prioridades

Evaluación de valor de negocio

- Impacto en ingresos
- Impacto en satisfacción de usuarios
- Impacto en competitividad
- Alineación con estrategia
- Retorno de inversión

Estimación de esfuerzo técnico

- Estimación de complejidad
- Estimación de tiempo
- Estimación de recursos
- Identificación de riesgos técnicos
- Validación de estimaciones

Gestión de dependencias

- Identificación de dependencias
- Análisis de impacto de dependencias
- Ordenamiento de tareas
- Gestión de camino crítico
- Comunicación de dependencias

Traducción de Necesidades de Negocio a Especificaciones Técnicas

Traducción de requisitos de negocio

- Comprensión de objetivos de negocio
- Identificación de requisitos técnicos
- Mapeo de requisitos de negocio a técnicos
- Validación de traducción
- Documentación de mapeo

Especificación técnica de funcionalidades

- Descripción de comportamiento técnico
- Definición de interfaces
- Especificación de datos
- Definición de flujos técnicos
- Documentación de especificaciones

Facilitación de comunicación entre equipos

- Comunicación entre negocio y desarrollo
- Clarificación de requisitos
- Resolución de malentendidos
- Documentación compartida
- Reuniones de alineación

Trazabilidad de requisitos

- Matriz de trazabilidad
- Vinculación de requisitos a código
- Vinculación de requisitos a pruebas
- Seguimiento de cambios
- Reportes de trazabilidad

Comunicación Efectiva con Stakeholders

Escucha activa y empatía

- Técnicas de escucha activa
- Comprensión de perspectivas diferentes
- Validación de comprensión
- Preguntas de clarificación
- Documentación de perspectivas

Rigor y precisión en documentación

- Trazabilidad de requisitos
- Completitud de criterios de aceptación
- Actualización de documentación
- Verificación de precisión
- Mantenimiento de documentación

Análisis objetivo basado en datos

- Recopilación de datos cuantitativos
- Análisis de múltiples fuentes
- Documentación de supuestos
- Presentación de opciones alternativas
- Evaluación de ventajas y desventajas

Negociación constructiva

- Presentación de opciones de alcance
- Evaluación de impacto en cronograma
- Escucha de preocupaciones

- Búsqueda de soluciones ganar-ganar
- Documentación de acuerdos

Habilidades actitudinales

- Escucha activa y comprensión empática de necesidades de stakeholders
- Rigor y precisión en documentación de requisitos
- Análisis objetivo basado en datos y evidencia
- Negociación constructiva de alcance y requisitos

Justificación

Este bloque agrupa competencias esenciales para la fase inicial de cualquier proyecto de desarrollo móvil. Las competencias C1 a C5 comparten el propósito común de traducir necesidades de negocio en especificaciones técnicas claras y documentadas. La coherencia funcional radica en que todas ellas requieren comunicación efectiva con stakeholders, análisis crítico de requisitos y documentación estructurada. Los entregables compatibles incluyen matriz de trazabilidad, especificaciones técnicas, historias de usuario y documentación de alcance. Las situaciones de evaluación realistas incluyen sesiones de recopilación de requisitos con clientes, análisis de documentación de negocio y validación de especificaciones con equipos técnicos.

Ubicación en la progresión

Este bloque constituye el fundamento de todo proyecto de desarrollo móvil, posicionándose al inicio del ciclo de vida de la aplicación. Los desarrolladores deben dominar estas competencias antes de pasar a fases de diseño y codificación, ya que requisitos mal especificados generan retrabajos costosos. El bloque integra tanto habilidades técnicas de análisis como habilidades blandas de comunicación, preparando al desarrollador para colaborar efectivamente con stakeholders no técnicos. La progresión natural lleva a los desarrolladores desde la comprensión de necesidades de negocio hacia la definición de arquitectura técnica, permitiendo que decisiones posteriores se basen en especificaciones sólidas y validadas.

Bloque 2 - Selección de Plataforma y Configuración del Entorno de Desarrollo

Objetivo pedagógico

Capacitar al desarrollador para evaluar y seleccionar plataformas de desarrollo móvil, configurar entornos de trabajo y gestionar dependencias técnicas de forma eficiente.

Actividades

Actividad 3.1 Evaluar y seleccionar plataformas objetivo para el desarrollo
Actividad 3.2 Seleccionar enfoque de desarrollo (nativo o multiplataforma)
Actividad 3.3 Configurar entorno de desarrollo e instalar dependencias

Competencias

Competencias	Indicadores
C7 - Evaluar y seleccionar plataforma de desarrollo (Android, iOS o multiplataforma) basándose en requisitos técnicos y restricciones de negocio	Realiza análisis comparativo de al menos 3 opciones de plataforma (nativa Android, nativa iOS, multiplataforma) con matriz de evaluación Documenta decisión de plataforma con justificación técnica basada en rendimiento, mantenibilidad, cuota de mercado y restricciones de dispositivos
C8 - Configurar entorno de desarrollo móvil instalando IDEs, emuladores y dependencias para preparar infraestructura de codificación	Instala y configura IDE (Android Studio o Xcode) con emuladores funcionales y dispositivos virtuales, validando con prueba de compilación exitosa Documenta procedimiento de configuración de entorno con pasos reproducibles y lista de verificación de validación
C9 - Gestionar dependencias y SDK del proyecto móvil resolviendo conflictos de compatibilidad para mantener estabilidad de compilación	Identifica y documenta todas las dependencias del proyecto con versiones compatibles, resolviendo al menos 3 conflictos de compatibilidad Implementa estrategia de actualización de dependencias con pruebas de regresión y documentación de cambios

Saberes asociados

Ecosistema Android y características técnicas

Arquitectura de Android

- Capas del sistema operativo Android
- Componentes principales (Activities, Services, Broadcast Receivers, Content Providers)
- Ciclo de vida de aplicaciones Android
- Permisos y seguridad en Android

Versiones y fragmentación de Android

- Historial de versiones de Android y características por versión
- Análisis de cuota de mercado por versión
- Compatibilidad hacia atrás y hacia adelante
- Restricciones de dispositivos por versión

Características de dispositivos Android

- Variedad de tamaños de pantalla y resoluciones
- Capacidades de procesamiento y memoria
- Sensores disponibles en dispositivos
- Restricciones de batería y conectividad

Ecosistema iOS y directrices de Apple

Arquitectura de iOS

- Capas del sistema operativo iOS
- Componentes principales de aplicaciones iOS
- Ciclo de vida de aplicaciones iOS
- Restricciones de seguridad y sandboxing

Directrices de Apple (HIG)

- Estándares de diseño de Apple
- Requisitos de funcionalidad y rendimiento
- Políticas de privacidad y seguridad
- Proceso de revisión de App Store

Dispositivos Apple y versiones

- Línea de productos Apple (iPhone, iPad)
- Versiones de iOS y características
- Compatibilidad de dispositivos con versiones
- Restricciones técnicas por modelo

Herramientas y entornos de desarrollo integrados

Android Studio

- Instalación y configuración de Android Studio
- Configuración de SDK y herramientas de compilación
- Creación y configuración de proyectos Android
- Uso de emulador de Android

Xcode

- Instalación y configuración de Xcode
- Configuración de SDK de iOS
- Creación y configuración de proyectos iOS
- Uso de simulador de iOS

Herramientas de compilación y gestión de dependencias

- Gradle para proyectos Android
- CocoaPods para proyectos iOS
- Configuración de build.gradle y Podfile
- Resolución de dependencias

Frameworks y enfoques de desarrollo multiplataforma

Desarrollo nativo vs multiplataforma

- Ventajas y desventajas de desarrollo nativo
- Ventajas y desventajas de frameworks multiplataforma
- Impacto en rendimiento y experiencia de usuario
- Consideraciones de mantenibilidad y escalabilidad

Frameworks multiplataforma principales

- React Native: características y limitaciones
- Flutter: características y limitaciones
- Xamarin: características y limitaciones

- Comparación de rendimiento y adopción

Evaluación de plataforma objetivo

- Análisis de requisitos técnicos del proyecto
- Evaluación de restricciones de negocio
- Análisis de cuota de mercado y audiencia
- Documentación de decisión de plataforma

Gestión de dependencias y resolución de conflictos

Identificación de dependencias

- Análisis de requisitos de librerías
- Evaluación de compatibilidad de versiones
- Documentación de dependencias necesarias
- Selección de librerías apropiadas

Resolución de conflictos de compatibilidad

- Identificación de conflictos de versiones
- Estrategias de resolución de conflictos
- Actualización segura de dependencias
- Prueba de compatibilidad después de cambios

Mantenimiento de dependencias

- Monitoreo de actualizaciones de librerías
- Evaluación de impacto de actualizaciones
- Documentación de cambios de dependencias
- Gestión de versiones de SDK

Validación y configuración de entorno de desarrollo

Configuración de emuladores y dispositivos virtuales

- Instalación de emuladores de Android
- Configuración de simuladores de iOS
- Creación de perfiles de dispositivos virtuales
- Optimización de rendimiento de emuladores

Conexión con dispositivos físicos

- Configuración de depuración USB en Android
- Configuración de conexión inalámbrica
- Instalación de aplicaciones en dispositivos físicos
- Validación de conectividad

Validación de entorno funcional

- Verificación de instalación de herramientas
- Prueba de compilación de proyecto base
- Validación de emuladores y dispositivos
- Documentación de procedimientos de configuración

Habilidades actitudinales

- Análisis objetivo de opciones técnicas basado en criterios cuantitativos
- Justificación transparente de decisiones técnicas
- Meticulosidad en configuración de entorno
- Proactividad en resolución de problemas técnicos

Justificación

Este bloque agrupa competencias críticas para preparar la infraestructura técnica del proyecto. Las competencias C7, C8 y C9 comparten el propósito común de establecer una base técnica sólida que permita desarrollo productivo. La coherencia funcional se evidencia en que todas requieren conocimiento profundo de ecosistemas móviles, capacidad de resolución de problemas técnicos y documentación clara de procedimientos. Los entregables compatibles incluyen matriz de evaluación de plataformas, documentación de configuración de entorno, lista de dependencias versionadas y procedimientos de resolución de conflictos. Las situaciones de evaluación realistas incluyen configuración de nuevo proyecto, resolución de conflictos de dependencias y validación de entorno funcional.

Ubicación en la progresión

Este bloque se posiciona inmediatamente después de la especificación de requisitos, ya que la selección de plataforma debe basarse en requisitos técnicos claramente definidos. Los desarrolladores deben dominar estas competencias antes de iniciar codificación, asegurando que el entorno está correctamente configurado y que todas las dependencias son compatibles. El bloque combina decisiones arquitectónicas de alto nivel con tareas técnicas operacionales, preparando al equipo para trabajar de forma eficiente. La progresión natural lleva desde la evaluación estratégica de plataformas hacia la configuración práctica de herramientas, permitiendo que el desarrollo comience con una base técnica estable.

Bloque 3 - Diseño de Arquitectura e Interfaz de Aplicaciones Móviles

Objetivo pedagógico

Capacitar al desarrollador para diseñar arquitectura de información, interfaces de usuario y patrones de navegación que sean escalables, accesibles y centrados en el usuario.

Actividades

Actividad 2.1 Diseñar arquitectura de información y crear prototipos de pantallas
Actividad 2.2 Validar wireframes y diseñar interfaces de usuario adaptativas
Actividad 2.3 Integrar principios de experiencia de usuario y diseño centrado en el usuario
Actividad 2.4 Crear componentes reutilizables y gestionar sistema de diseño

Competencias

Competencias	Indicadores
C6 - Diseñar arquitectura de información y modelado de datos para estructurar la aplicación móvil de forma escalable y mantenible	Elabora diagrama de arquitectura de información que incluye entidades, relaciones y flujos de datos con documentación de decisiones Crea modelo de datos normalizado con especificación de restricciones, índices y estrategia de migración
C10 - Implementar lógica de negocio aplicando patrones de arquitectura (MVC, MVP, MVVM) para asegurar separación de responsabilidades y mantenibilidad	Desarrolla al menos 5 módulos de lógica de negocio siguiendo patrón arquitectónico seleccionado con separación clara de capas Documenta estructura de clases y responsabilidades con diagramas UML y código comentado que explica decisiones de diseño
C11 - Implementar navegación entre pantallas de la aplicación móvil gestionando flujos de usuario y paso de parámetros para crear experiencia coherente	Implementa sistema de navegación que soporta al menos 10 pantallas diferentes con gestión correcta de historial y parámetros Valida flujos de navegación en múltiples escenarios (entrada normal, regreso, cambio de orientación) documentando casos de prueba
C12 - Desarrollar interfaces de usuario móviles siguiendo directrices de diseño (Material Design, HIG) para asegurar consistencia y accesibilidad	Implementa al menos 15 componentes visuales siguiendo Material Design (Android) o Human Interface Guidelines (iOS) con validación de accesibilidad Realiza pruebas de accesibilidad con herramientas de validación (Accessibility Scanner, Accessibility Inspector) documentando conformidad
C13 - Construir layouts adaptativos y responsivos para múltiples tamaños de pantalla utilizando unidades relativas y restricciones de layout	Diseña y valida layouts que se adaptan correctamente a al menos 6 tamaños de pantalla diferentes (teléfono, tablet, orientaciones) Implementa sistema de restricciones o unidades relativas que mantiene proporciones y espaciado en todos los dispositivos

Competencias	Indicadores
C14 - Crear componentes reutilizables de interfaz documentando API y manteniendo biblioteca centralizada para optimizar desarrollo y consistencia	Desarrolla al menos 8 componentes reutilizables con documentación clara de propiedades, métodos y ejemplos de uso Mantiene biblioteca de componentes versionada con proceso de revisión y actualización documentado
C15 - Gestionar estado de la aplicación móvil implementando patrones de gestión (Redux, BLoC, Provider) para sincronizar datos entre componentes	Implementa gestión de estado en al menos 3 flujos complejos de la aplicación utilizando patrón seleccionado con validación de sincronización Documenta flujos de estado con diagramas de transición y pruebas que validan cambios de estado en escenarios normales y excepcionales
C16 - Integrar patrones de diseño centrado en el usuario recopilando feedback y validando interacciones para iterar basándose en necesidades reales	Realiza al menos 2 sesiones de recopilación de feedback con usuarios reales documentando insights y cambios solicitados Implementa cambios basados en feedback y valida mejoras con pruebas de usabilidad adicionales
C17 - Diseñar prototipos de pantallas móviles de baja y alta fidelidad utilizando herramientas de diseño para validar conceptos antes de desarrollo	Crea prototipos de al menos 10 pantallas principales en herramienta de diseño (Figma, Sketch) con flujos de interacción funcionales Genera especificaciones de diseño (dimensiones, colores, tipografía) exportables para equipo de desarrollo
C18 - Validar wireframes y prototipos con stakeholders obteniendo aprobación formal antes de iniciar desarrollo de funcionalidades	Presenta prototipos a al menos 3 stakeholders diferentes documentando feedback recibido y cambios solicitados Obtiene aprobación formal documentada de diseño antes de iniciar desarrollo de cada módulo principal

Saberes asociados

Arquitectura de Información y Modelado de Datos

Diseño de Arquitectura de Información

- Estructuración lógica de datos
- Organización de entidades y relaciones
- Patrones de arquitectura de información
- Documentación de arquitectura

Modelado de Datos

- Definición de entidades y atributos
- Relaciones entre entidades
- Restricciones de integridad
- Normalización de datos

Diseño de Flujos de Navegación

- Mapeo de pantallas y transiciones
- Flujos de usuario
- Gestión del historial de navegación
- Paso de parámetros entre pantallas

Patrones Arquitectónicos y Separación de Responsabilidades

Patrones MVC, MVP, MVVM

- Arquitectura Modelo-Vista-Controlador
- Arquitectura Modelo-Vista-Presentador

- Arquitectura Modelo-Vista-ViewModel
- Comparación de patrones

Separación de Responsabilidades

- Principios SOLID
- Cohesión y acoplamiento
- Capas de aplicación
- Inyección de dependencias

Gestión de Estado

- Patrones Redux, BLoC, Provider
- Sincronización de datos entre componentes
- Optimización de flujos de estado
- Manejo de estado compartido

Diseño de Interfaz de Usuario y Experiencia

Directrices de Diseño

- Material Design de Google
- Human Interface Guidelines de Apple
- Principios de usabilidad
- Accesibilidad en interfaces móviles

Diseño Responsivo y Adaptativo

- Layouts adaptativos
- Unidades relativas (dp, sp, porcentajes)
- Restricciones de layout
- Prueba en múltiples tamaños de pantalla

Componentes Reutilizables

- Identificación de componentes
- Abstracción de componentes
- Documentación de API
- Biblioteca centralizada de componentes

Diseño Centrado en Usuario

- Metodologías UCD
- Recopilación de feedback
- Validación de interacciones
- Iteración basada en feedback

Prototipado y Validación de Diseño

Diseño de Prototipos

- Prototipos de baja fidelidad
- Prototipos de alta fidelidad
- Herramientas de diseño (Figma, Sketch)
- Representación de flujos de interacción

Validación con Stakeholders

- Presentación de wireframes
- Recopilación de feedback
- Documentación de cambios
- Obtención de aprobación formal

Pruebas de Usabilidad

- Evaluación de accesibilidad
- Pruebas con usuarios reales

- Identificación de problemas de usabilidad
- Iteración de diseño

Implementación de Componentes y Navegación

Desarrollo de Componentes Visuales

- Creación de elementos de interfaz
- Implementación de Material Design
- Implementación de HIG
- Accesibilidad en componentes

Implementación de Navegación

- Componentes de navegación nativos
- Gestión del historial
- Paso de parámetros
- Transiciones entre pantallas

Animaciones e Interacciones

- Animaciones de transición
- Micro-interacciones
- APIs de animación nativas
- Optimización de rendimiento

Postura Profesional en Diseño

Consistencia y Rigor

- Aplicación consistente de patrones
- Documentación de excepciones
- Revisión de código
- Adherencia a estándares

Anticipación y Escalabilidad

- Diseño con extensibilidad
- Evitar acoplamiento innecesario
- Documentación de puntos de extensión
- Preparación para cambios futuros

Atención al Detalle

- Validación visual
- Consistencia de diseño
- Prueba en múltiples dispositivos
- Accesibilidad e inclusión

Apertura a Feedback

- Solicitud de feedback específico
- Escucha activa
- Reconocimiento de problemas reales
- Mejora continua

Habilidades actitudinales

- Consistencia en aplicación de patrones arquitectónicos
- Anticipación de cambios futuros en diseño
- Atención al detalle visual y de usabilidad
- Orientación hacia accesibilidad e inclusión
- Apertura a feedback crítico de usuarios y stakeholders

Justificación

Este bloque agrupa competencias que abarcan tanto el diseño conceptual como la implementación de arquitectura e interfaz. Las competencias C6 a C18 comparten el propósito común de crear una estructura técnica y visual que sea mantenible, accesible y alineada con necesidades de usuarios. La coherencia funcional se evidencia en que todas requieren pensamiento sistémico, comprensión de principios de diseño y capacidad de traducir requisitos en soluciones concretas. Los entregables compatibles incluyen diagramas de arquitectura, prototipos de interfaz, especificaciones de diseño, componentes reutilizables y documentación de patrones de navegación. Las situaciones de evaluación realistas incluyen diseño de nueva aplicación, validación de prototipos con usuarios, implementación de componentes visuales y pruebas de accesibilidad.

Ubicación en la progresión

Este bloque se posiciona después de la especificación de requisitos y selección de plataforma, integrando decisiones técnicas previas en una arquitectura coherente. Los desarrolladores deben dominar estas competencias antes de iniciar desarrollo de funcionalidades, asegurando que la estructura base es sólida y escalable. El bloque combina pensamiento de diseño centrado en usuario con principios de arquitectura técnica, preparando al equipo para implementar funcionalidades de forma consistente. La progresión natural lleva desde el diseño conceptual de arquitectura hacia la implementación de componentes visuales, permitiendo que desarrollo posterior se realice de forma ordenada y predecible.

Bloque 4 - Desarrollo de Funcionalidades y Lógica de Negocio

Objetivo pedagógico

Capacitar al desarrollador para implementar lógica de negocio, persistencia de datos y comunicación con servicios externos de forma segura y mantenible.

Actividades

Actividad 4.1 Programar lógica de negocio e implementar navegación
Actividad 4.2 Desarrollar interfaces de usuario y layouts adaptativos
Actividad 4.3 Crear componentes reutilizables y gestionar estado de la aplicación
Actividad 4.4 Implementar formularios, validaciones y animaciones

Competencias

Competencias	Indicadores
C19 - Implementar persistencia local de datos utilizando bases de datos móviles (SQLite, Room, Core Data) con gestión de migraciones	Diseña e implementa esquema de base de datos con al menos 5 tablas, índices y restricciones de integridad referencial Implementa sistema de migraciones que permite actualizar esquema sin pérdida de datos en al menos 3 versiones de aplicación
C20 - Conectar aplicación móvil con APIs externas implementando clientes HTTP y gestionando errores de red para asegurar comunicación confiable	Implementa cliente HTTP que realiza al menos 10 tipos diferentes de llamadas a API con manejo de errores y reintentos Valida manejo de fallos de conectividad (timeout, sin conexión, respuestas incompletas) con pruebas de escenarios offline
C21 - Consumir servicios web REST desde aplicación móvil implementando métodos HTTP y abstrayendo lógica de red en repositorios	Implementa repositorios que abstraen llamadas REST para al menos 5 recursos diferentes con métodos GET, POST, PUT, DELETE Valida que respuestas HTTP se procesan correctamente con códigos de estado (200, 201, 400, 401, 404, 500) y manejo de errores
C22 - Procesar respuestas JSON de servicios remotos deserializando datos y mapeándolos a modelos de aplicación con validación	Implementa deserializadores JSON para al menos 8 estructuras de datos diferentes con manejo de campos opcionales y anidados Valida datos recibidos contra esquema esperado y maneja respuestas incompletas o malformadas con mensajes de error informativos
C27 - Desarrollar formularios de entrada de datos con validación, gestión de teclado virtual y navegación accesible entre campos	Implementa al menos 3 formularios complejos con validación en tiempo real, gestión de teclado virtual y navegación fluida Valida accesibilidad de formularios con herramientas de validación y pruebas con usuarios con discapacidades
C28 - Validar campos y entradas del usuario implementando validaciones cliente y servidor con mensajes de error contextuales	Implementa validaciones para al menos 10 tipos de campos diferentes (email, teléfono, fecha, contraseña) con mensajes de error claros Valida que validaciones cliente se complementan con validaciones servidor para seguridad

Saberes asociados

Persistencia de Datos en Aplicaciones Móviles

Bases de Datos Locales

- SQLite en Android
- Room ORM para Android
- Core Data en iOS
- Esquemas de datos
- Relaciones entre entidades
- Índices y optimización

Migraciones de Datos

- Versionado de esquema
- Estrategias de migración
- Preservación de datos de usuario
- Rollback de migraciones
- Testing de migraciones

Almacenamiento Seguro

- Keystore de Android
- Keychain de iOS
- Cifrado de datos locales
- Protección de credenciales
- Gestión de secretos

Integración con Servicios Web y APIs

Clientes HTTP

- Retrofit para Android
- Alamofire para iOS
- Dio para Flutter
- Configuración de clientes
- Interceptores y middleware
- Gestión de timeouts

Consumo de APIs REST

- Métodos HTTP (GET, POST, PUT, DELETE)
- Parámetros de solicitud
- Headers y autenticación
- Códigos de estado HTTP
- Manejo de errores de red
- Reintentos y circuit breaker

Procesamiento de Datos

- Deserialización JSON
- Mapeo objeto-relacional
- Validación de datos
- Transformación de respuestas
- Manejo de datos incompletos
- Conversión de tipos

Validación de Entrada de Usuario

Validaciones Cliente

- Validación de formato
- Validación de rango
- Validación de longitud
- Validación de tipo de dato
- Validación de patrones regex

- Validación en tiempo real

Validaciones Servidor

- Validación de negocio
- Validación de integridad referencial
- Validación de duplicados
- Validación de autorización
- Respuestas de error estructuradas

Mensajes de Error

- Mensajes contextuales
- Internacionalización de mensajes
- Guía de corrección
- Accesibilidad en mensajes
- Logging de errores de validación

Manejo de Conectividad y Resiliencia de Red

Detección de Conectividad

- Verificación de estado de red
- Detección de cambios de conectividad
- Diferenciación entre WiFi y datos móviles
- Monitoreo de calidad de conexión

Estrategias de Sincronización

- Sincronización offline-first
- Cola de operaciones pendientes
- Resolución de conflictos
- Caché local
- Sincronización incremental

Manejo de Fallos

- Reintentos exponenciales
- Timeouts configurables
- Fallback a datos en caché
- Notificación al usuario
- Recuperación automática

Patrones de Arquitectura en Aplicaciones Móviles

Patrones MVC, MVP, MVVM

- Separación de responsabilidades
- Flujo de datos
- Comunicación entre capas
- Testing de componentes
- Ventajas y desventajas de cada patrón

Capas de Aplicación

- Capa de presentación
- Capa de lógica de negocio
- Capa de acceso a datos
- Capa de servicios
- Inyección de dependencias

Patrones de Integración

- Patrón repositorio
- Patrón adaptador
- Patrón observador

- Patrón factory
- Patrón singleton

Seguridad en Manejo de Datos Móviles

Protección de Datos Sensibles

- Cifrado de datos en tránsito
- Cifrado de datos en reposo
- Certificados SSL/TLS
- Validación de certificados
- Pinning de certificados

Vulnerabilidades Comunes

- Inyección de SQL
- Inyección de código
- Exposición de datos sensibles
- Almacenamiento inseguro
- Comunicación insegura

Buenas Prácticas de Seguridad

- Validación de entrada
- Sanitización de datos
- Principio de menor privilegio
- Auditoría de seguridad
- Gestión de secretos

Habilidades actitudinales

- Cuidado con integridad y validez de datos
- Resiliencia ante fallos de red y conectividad
- Rigor en validación de entrada de usuario
- Vigilancia de seguridad en manejo de datos

Justificación

Este bloque agrupa competencias esenciales para la implementación de funcionalidades principales de la aplicación. Las competencias C19 a C28 comparten el propósito común de traducir requisitos en código funcional que interactúa correctamente con datos y servicios externos. La coherencia funcional se evidencia en que todas requieren comprensión de patrones de integración, gestión de datos y manejo de errores. Los entregables compatibles incluyen módulos de lógica de negocio, capas de acceso a datos, clientes HTTP y validadores de entrada. Las situaciones de evaluación realistas incluyen implementación de nuevas funcionalidades, integración con APIs externas, manejo de fallos de conectividad y validación de datos de usuario.

Ubicación en la progresión

Este bloque se posiciona después del diseño de arquitectura, implementando la estructura definida en fases anteriores. Los desarrolladores deben dominar estas competencias para traducir especificaciones técnicas en código funcional que cumple requisitos de negocio. El bloque combina patrones de arquitectura con técnicas de integración de datos, preparando al equipo para implementar funcionalidades complejas de forma confiable. La progresión natural lleva desde la implementación de lógica de negocio básica hacia la integración con servicios externos, permitiendo que funcionalidades se construyan incrementalmente.

Bloque 5 - Integración de Servicios Avanzados y Funcionalidades Especializadas

Objetivo pedagógico

Capacitar al desarrollador para integrar servicios avanzados como autenticación, notificaciones push y seguridad, implementando funcionalidades especializadas que mejoran experiencia y confiabilidad.

Actividades

Actividad 5.1 Implementar persistencia local y conectar con APIs externas
Actividad 5.2 Consumir servicios REST y procesar respuestas JSON
Actividad 5.3 Integrar autenticación, sesiones y notificaciones push
Actividad 5.4 Integrar funcionalidades de hardware del dispositivo

Competencias

Competencias	Indicadores
C23 - Integrar autenticación de usuarios implementando flujos OAuth, Firebase Auth o autenticación personalizada con recuperación de contraseña	Implementa al menos 2 métodos de autenticación diferentes (OAuth, Firebase, credenciales) con flujos de registro y recuperación de contraseña Valida que credenciales se transmiten de forma segura (HTTPS, hash) y que sesiones se cierran correctamente
C24 - Gestionar sesiones y tokens de acceso almacenando de forma segura y renovando automáticamente para mantener autenticación persistente	Implementa almacenamiento seguro de tokens (Keychain, Keystore) con cifrado y protección contra acceso no autorizado Valida renovación automática de tokens JWT antes de expiración y cierre de sesión por timeout con pruebas de escenarios de expiración
C25 - Implementar notificaciones push integrando Firebase Cloud Messaging o Apple Push Notification service con gestión de permisos	Integra servicio de notificaciones push con manejo correcto de permisos en primer plano y segundo plano Valida recepción y procesamiento de notificaciones en múltiples escenarios (aplicación abierta, cerrada, en segundo plano)
C26 - Añadir animaciones e interacciones en interfaz móvil utilizando APIs nativas optimizando rendimiento para mantener fluidez	Implementa al menos 8 animaciones diferentes (transiciones, micro-interacciones) con validación de rendimiento (60 FPS) Documenta decisiones de animación y valida que no impactan negativamente en consumo de batería o memoria
C34 - Aplicar principios de seguridad en desarrollo identificando vulnerabilidades OWASP Mobile e implementando cifrado y almacenamiento seguro	Realiza análisis de seguridad identificando al menos 5 vulnerabilidades potenciales según OWASP Mobile Top 10 Implementa cifrado para datos sensibles y almacenamiento seguro de credenciales con validación de conformidad

Saberes asociados

Autenticación y Gestión de Sesiones Seguras

Flujos de Autenticación

- Implementación de OAuth 2.0
- Integración de Firebase Authentication
- Autenticación personalizada con recuperación de contraseña
- Flujos de login y registro
- Manejo de múltiples métodos de autenticación

Gestión de Tokens y Sesiones

- Almacenamiento seguro de tokens en Keystore/Keychain
- Gestión de JWT (JSON Web Tokens)
- Renovación automática de tokens
- Cierre de sesión por expiración
- Protección contra ataques de token

Seguridad en Autenticación

- Hash seguro de contraseñas
- Protección contra ataques de fuerza bruta
- Validación de credenciales
- Manejo seguro de datos sensibles

Notificaciones Push y Mensajería

Integración de Servicios de Notificaciones

- Firebase Cloud Messaging (FCM) para Android
- Apple Push Notification service (APNs) para iOS
- Configuración de certificados y credenciales
- Registro de dispositivos para notificaciones

Gestión de Notificaciones

- Manejo de notificaciones en primer plano
- Manejo de notificaciones en segundo plano
- Gestión de permisos de notificaciones
- Acciones personalizadas en notificaciones
- Gestión de canales de notificación

Experiencia de Usuario en Notificaciones

- Diseño de mensajes claros y concisos
- Timing y frecuencia de notificaciones
- Personalización de notificaciones
- Análisis de engagement de notificaciones

Seguridad Móvil y Vulnerabilidades OWASP

Vulnerabilidades OWASP Mobile Top 10

- Inyección de código
- Almacenamiento inseguro de datos
- Comunicación insegura
- Autenticación débil
- Criptografía insuficiente
- Autorización deficiente
- Configuración de seguridad deficiente
- Exfiltración de datos sensibles

Implementación de Cifrado

- Cifrado de datos en tránsito (TLS/SSL)
- Cifrado de datos en reposo
- Algoritmos criptográficos seguros
- Gestión de claves criptográficas

- Validación de certificados SSL

Almacenamiento Seguro de Datos

- Almacenamiento seguro en Keystore (Android)
- Almacenamiento seguro en Keychain (iOS)
- Encriptación de bases de datos locales
- Protección de datos sensibles en memoria
- Limpieza segura de datos

Revisión de Seguridad de Código

- Análisis estático de seguridad
- Identificación de vulnerabilidades comunes
- Pruebas de penetración
- Auditoría de seguridad
- Documentación de hallazgos de seguridad

Animaciones e Interacciones Avanzadas

Implementación de Animaciones

- Animaciones de transición entre pantallas
- Micro-interacciones y feedback visual
- Animaciones de propiedades (escala, rotación, opacidad)
- Animaciones de lista y scroll
- Animaciones complejas y secuenciadas

APIs de Animación Nativas

- Android Animation Framework
- iOS Core Animation
- Flutter Animation APIs
- Uso de herramientas de animación especializadas

Optimización de Rendimiento

- Mantenimiento de 60 FPS
- Reducción de consumo de batería
- Optimización de memoria en animaciones
- Profiling de rendimiento de animaciones
- Alternativas de bajo rendimiento

Experiencia de Usuario en Interacciones

- Feedback visual inmediato
- Transiciones suaves y naturales
- Accesibilidad en animaciones
- Respeto a preferencias de movimiento del usuario

Integración de Servicios Avanzados

Patrones de Integración

- Arquitectura de servicios en aplicaciones móviles
- Patrones de comunicación asincrónica
- Manejo de dependencias entre servicios
- Integración de múltiples servicios

Gestión de Permisos del Sistema

- Solicitud de permisos en tiempo de ejecución
- Manejo de permisos denegados
- Permisos peligrosos vs normales
- Justificación de permisos a usuarios

Manejo de Errores en Servicios

- Recuperación de fallos de servicio
- Reintentos inteligentes
- Fallback a funcionalidad alternativa
- Comunicación clara de errores a usuarios

Postura y Responsabilidad en Seguridad

Vigilancia de Seguridad

- Revisión proactiva de código para vulnerabilidades
- Mantenimiento de conocimiento actualizado de amenazas
- Implementación de controles de seguridad recomendados
- Auditoría regular de seguridad

Responsabilidad en Datos Sensibles

- Nunca hardcodear credenciales o claves
- Almacenamiento en sistemas seguros
- Limitación de acceso a claves de firma
- Auditoría de acceso a información sensible

Sensibilidad a Experiencia del Usuario

- Prueba desde perspectiva del usuario
- Identificación de puntos de fricción
- Validación de mejora en experiencia
- Equilibrio entre seguridad y usabilidad

Habilidades actitudinales

- Vigilancia constante de seguridad en integración de servicios
- Responsabilidad en manejo de datos sensibles
- Sensibilidad a experiencia del usuario en interacciones
- Atención a detalles de accesibilidad en funcionalidades avanzadas

Justificación

Este bloque agrupa competencias para integración de servicios complejos que requieren conocimiento especializado. Las competencias C23 a C34 comparten el propósito común de añadir funcionalidades avanzadas que mejoran seguridad, experiencia de usuario y confiabilidad de la aplicación. La coherencia funcional se evidencia en que todas requieren integración con servicios externos, manejo de permisos del sistema y consideraciones de seguridad. Los entregables compatibles incluyen flujos de autenticación, sistemas de notificaciones, análisis de seguridad y componentes de animación. Las situaciones de evaluación realistas incluyen integración de OAuth, implementación de notificaciones push, análisis de vulnerabilidades OWASP y optimización de animaciones.

Ubicación en la progresión

Este bloque se posiciona después de la implementación de funcionalidades básicas, añadiendo capas de sofisticación que mejoran la aplicación. Los desarrolladores deben dominar competencias previas antes de abordar estos servicios avanzados, ya que requieren comprensión profunda de arquitectura y seguridad. El bloque combina integración técnica con consideraciones de experiencia de usuario, preparando al equipo para implementar funcionalidades que diferencia la aplicación. La progresión natural lleva desde autenticación básica hacia sistemas complejos de notificaciones y seguridad, permitiendo que aplicación evolucione hacia mayor sofisticación.

Bloque 6 - Pruebas, Depuración y Optimización de Aplicaciones Móviles

Objetivo pedagógico

Capacitar al desarrollador para escribir pruebas automatizadas, depurar errores complejos y validar compatibilidad en múltiples dispositivos, asegurando calidad y confiabilidad.

Actividades

Actividad 6.1 Ejecutar pruebas funcionales y de compatibilidad
Actividad 6.2 Escribir pruebas unitarias y automatizadas de interfaz
Actividad 6.3 Depurar errores, gestionar excepciones y optimizar rendimiento
Actividad 6.4 Monitorizar incidencias y corregir errores en producción

Competencias

Competencias	Indicadores
C29 - Gestionar errores y excepciones de ejecución implementando manejo robusto con logging y análisis para mejorar confiabilidad	Implementa manejo de excepciones en al menos 20 puntos críticos de la aplicación con mensajes de error comprensibles para usuarios Configura sistema de logging que registra errores con contexto suficiente para análisis y debugging
C30 - Depurar fallos y errores de aplicación móvil utilizando herramientas de depuración, análisis de logs y reproducción de errores	Utiliza herramientas de depuración (debugger, profiler, logcat) para identificar y resolver al menos 10 errores complejos Documenta proceso de reproducción de errores y soluciones implementadas para prevenir recurrencia
C31 - Escribir pruebas unitarias para lógica de negocio utilizando frameworks de testing (JUnit, XCTest, Flutter Test) con cobertura adecuada	Implementa pruebas unitarias para al menos 15 funciones críticas de lógica de negocio con cobertura mínima del 80% Valida que pruebas son independientes, reproducibles y ejecutables en pipeline de integración continua
C32 - Desarrollar pruebas automatizadas de interfaz de usuario implementando escenarios de prueba con herramientas de testing (Espresso, XCUITest, Appium)	Implementa pruebas automatizadas para al menos 10 flujos de usuario críticos validando interacciones y resultados esperados Integra pruebas en pipeline CI/CD con ejecución automática en cada compilación
C33 - Ejecutar pruebas de compatibilidad en matriz de dispositivos documentando problemas y priorizando defectos para asegurar funcionamiento multiplataforma	Realiza pruebas en al menos 8 dispositivos diferentes (variedad de tamaños, versiones de SO, fabricantes) documentando resultados Prioriza defectos encontrados según severidad e impacto en usuarios, creando plan de corrección

Saberes asociados

Marcos de trabajo de pruebas automatizadas

Pruebas unitarias

- Diseño de casos de prueba unitarios
- Implementación con JUnit para Android
- Implementación con XCTest para iOS
- Implementación con Flutter Test
- Cobertura de pruebas y métricas
- Mocking y stubbing de dependencias

Pruebas de interfaz de usuario

- Implementación con Espresso para Android
- Implementación con XCUITest para iOS
- Implementación con Appium multiplataforma
- Definición de escenarios de prueba
- Automatización de interacciones de usuario
- Validación de elementos visuales

Pruebas de compatibilidad

- Planificación de matriz de dispositivos
- Ejecución en emuladores y dispositivos físicos
- Documentación de problemas de compatibilidad
- Priorización de defectos
- Análisis de resultados de pruebas

Técnicas de depuración y análisis de errores

Herramientas de depuración

- Uso de debuggers en Android Studio
- Uso de debuggers en Xcode
- Inspección de estado de aplicación
- Ejecución paso a paso de código
- Puntos de ruptura y condicionales

Análisis de errores y logs

- Análisis de trazas de error (stack traces)
- Revisión de logs de aplicación
- Análisis de volcados de memoria
- Identificación de patrones de errores
- Herramientas de análisis de crashs

Reproducción y resolución de errores

- Reproducción sistemática de errores
- Documentación de pasos de reproducción
- Validación de soluciones
- Prevención de regresiones

Gestión de errores y excepciones

Manejo de excepciones

- Captura y procesamiento de excepciones
- Jerarquía de excepciones
- Manejo de errores de red
- Manejo de fallos de conectividad
- Recuperación de errores

Logging y monitoreo

- Implementación de sistemas de logging
- Niveles de log (debug, info, warning, error)
- Análisis de logs para diagnóstico

- Monitoreo de errores en producción
- Herramientas de análisis de crashes

Mensajes de error

- Diseño de mensajes de error comprensibles
- Mensajes contextuales para usuarios
- Documentación de códigos de error
- Internacionalización de mensajes de error

Optimización y rendimiento de aplicaciones móviles

Análisis de rendimiento

- Profiling de CPU y memoria
- Análisis de consumo de batería
- Medición de tiempo de respuesta
- Identificación de cuellos de botella
- Herramientas de profiling

Optimización de código

- Optimización de algoritmos
- Gestión eficiente de memoria
- Caché y reutilización de datos
- Lazy loading y carga progresiva
- Optimización de consultas de base de datos

Optimización de interfaz

- Optimización de animaciones
- Renderizado eficiente de vistas
- Gestión de recursos gráficos
- Reducción de consumo de batería
- Mejora de fluidez (FPS)

Aseguramiento de calidad y validación

Estrategias de testing

- Pirámide de testing
- Testing en diferentes niveles
- Cobertura de código
- Criterios de aceptación
- Métricas de calidad

Validación de compatibilidad

- Matriz de dispositivos
- Versiones de sistema operativo
- Variabilidad de hardware
- Resoluciones de pantalla
- Configuraciones de red

Documentación de defectos

- Reporte de bugs
- Clasificación de severidad
- Trazabilidad de defectos
- Ciclo de vida de defectos
- Análisis de tendencias

Pruebas de seguridad y vulnerabilidades

Identificación de vulnerabilidades

- Vulnerabilidades OWASP Mobile Top 10
- Inyección de código
- Almacenamiento inseguro de datos
- Transmisión insegura de datos
- Autenticación débil

Revisión de seguridad de código

- Análisis estático de seguridad
- Revisión manual de código
- Identificación de patrones inseguros
- Validación de entrada
- Gestión segura de credenciales

Pruebas de penetración

- Pruebas de autenticación
- Pruebas de autorización
- Pruebas de cifrado
- Pruebas de almacenamiento de datos
- Análisis de comunicación de red

Habilidades actitudinales

- Mentalidad de calidad y rigor en testing
- Rigor en investigación de errores
- Proactividad en prevención de errores
- Documentación clara de pruebas y resultados

Justificación

Este bloque agrupa competencias esenciales para asegurar calidad de la aplicación antes de publicación. Las competencias C29 a C33 comparten el propósito común de identificar y resolver problemas, validar funcionamiento correcto y documentar calidad. La coherencia funcional se evidencia en que todas requieren pensamiento crítico, uso de herramientas de testing y capacidad de análisis sistemático. Los entregables compatibles incluyen suites de pruebas unitarias, pruebas automatizadas de interfaz, reportes de compatibilidad y análisis de defectos. Las situaciones de evaluación realistas incluyen escritura de pruebas para nuevas funcionalidades, depuración de errores complejos, ejecución de pruebas en matriz de dispositivos y análisis de logs de error.

Ubicación en la progresión

Este bloque se posiciona después de la implementación de funcionalidades, validando que todo funciona correctamente antes de publicación. Los desarrolladores deben dominar estas competencias para asegurar que aplicación cumple requisitos de calidad y funciona en múltiples dispositivos. El bloque combina técnicas de testing automatizado con depuración manual, preparando al equipo para identificar y resolver problemas de forma eficiente. La progresión natural lleva desde pruebas unitarias básicas hacia pruebas de compatibilidad complejas, permitiendo que calidad se valide de forma sistemática.

Bloque 7 - Publicación, Actualización y Mantenimiento de Aplicaciones Móviles

Objetivo pedagógico

Capacitar al desarrollador para preparar compilaciones de producción, gestionar claves de firma y publicar aplicaciones en tiendas de aplicaciones de forma segura.

Actividades

Actividad 7.1 Preparar compilaciones de producción y generar artefactos
Actividad 7.2 Publicar aplicaciones en tiendas móviles
Actividad 7.3 Actualizar aplicaciones y mantener compatibilidad con nuevas versiones

Competencias

Competencias	Indicadores
C35 - Preparar compilaciones para publicación configurando parámetros de producción, generando artefactos y gestionando claves de firma	Configura compilación de producción con optimizaciones (minificación, ofuscación) y genera artefactos distribuidores (APK, IPA) Gestiona claves de firma de forma segura con documentación de procedimiento de almacenamiento y acceso

Saberes asociados

Configuración de Compilaciones de Producción

Parámetros de compilación de producción

- Habilitación de optimizaciones de compilador
- Deshabilitación de código de depuración
- Configuración de niveles de optimización
- Eliminación de símbolos de depuración
- Configuración de ProGuard/R8 para ofuscación

Generación de artefactos distribuidores

- Generación de APK para Android
- Generación de AAB (Android App Bundle)
- Generación de IPA para iOS
- Configuración de arquitecturas de destino
- Validación de artefactos generados

Validación de compilaciones

- Prueba de artefactos antes de publicación
- Verificación de integridad de paquetes
- Validación de dependencias incluidas
- Prueba de funcionalidades críticas en artefacto final

Gestión Segura de Claves de Firma

Almacenamiento seguro de credenciales

- Almacenamiento en sistemas seguros (vaults, gestores de secretos)
- Protección contra acceso no autorizado
- Cifrado de claves en reposo
- Evitar hardcoding de credenciales
- Uso de variables de entorno seguras

Procedimientos de firma de aplicaciones

- Firma de APK con keystore Android
- Firma de IPA con certificados Apple
- Configuración de certificados de distribución
- Renovación de certificados antes de expiración
- Validación de firma de aplicación

Control de acceso a claves

- Limitación de acceso a personas autorizadas
- Auditoría de acceso a claves
- Documentación de procedimientos de acceso
- Rotación de claves según políticas de seguridad

Cumplimiento de Requisitos de Tiendas de Aplicaciones

Requisitos de Google Play Store

- Políticas de contenido y privacidad
- Requisitos de permisos y seguridad
- Directrices de calidad de aplicación
- Requisitos de accesibilidad
- Políticas de datos de usuario

Requisitos de Apple App Store

- Directrices de revisión de App Store
- Requisitos de privacidad y seguridad
- Directrices de diseño de interfaz
- Requisitos de funcionalidad
- Políticas de contenido

Documentación para publicación

- Descripción de aplicación y cambios
- Capturas de pantalla y previsualizaciones
- Notas de versión
- Información de privacidad
- Clasificación de contenido

Procesos de Publicación y Distribución

Publicación en tiendas de aplicaciones

- Carga de artefactos en consolas de desarrollador
- Configuración de información de aplicación
- Programación de lanzamiento
- Gestión de versiones y actualizaciones
- Monitoreo de estado de publicación

Distribución alternativa

- Distribución directa de APK
- Distribución mediante TestFlight (iOS)
- Distribución mediante Google Play Beta
- Distribución interna a equipos

Gestión de versiones

- Versionado semántico
- Gestión de números de versión
- Documentación de cambios entre versiones
- Planificación de actualizaciones

Seguridad en Procesos de Publicación

Responsabilidad en manejo de credenciales

- Nunca compartir claves de firma
- Documentación de acceso a credenciales
- Auditoría de quién accede a claves
- Revocación de acceso cuando personal se va

Vigilancia de seguridad

- Revisión de código antes de publicación
- Validación de que no hay datos sensibles en artefactos
- Verificación de que dependencias son seguras
- Análisis de vulnerabilidades antes de publicación

Cumplimiento normativo

- Cumplimiento de regulaciones de privacidad
- Cumplimiento de requisitos de tiendas
- Documentación de cumplimiento
- Auditoría de conformidad

Mantenimiento y Actualización de Aplicaciones

Gestión de actualizaciones

- Planificación de ciclos de actualización
- Gestión de compatibilidad hacia atrás
- Deprecación de funcionalidades antiguas
- Comunicación de cambios a usuarios

Monitoreo post-publicación

- Monitoreo de crashes y errores
- Análisis de métricas de uso
- Recopilación de feedback de usuarios
- Identificación de problemas en producción

Ciclo de vida de aplicación

- Soporte de versiones antiguas
- Planificación de fin de vida
- Migración de usuarios a nuevas versiones
- Archivado de versiones antiguas

Habilidades actitudinales

- Responsabilidad en manejo de claves de firma
- Meticulosidad en preparación de compilación
- Cumplimiento de requisitos de tiendas

Justificación

Este bloque agrupa competencias críticas para la fase final de desarrollo antes de publicación. La competencia C35 abarca el propósito común de preparar aplicación para distribución en tiendas oficiales. La coherencia funcional se evidencia en que requiere conocimiento de procesos de compilación, gestión segura

de credenciales y cumplimiento de requisitos de tiendas. Los entregables compatibles incluyen compilaciones optimizadas, artefactos distribuidores (APK, IPA, AAB) y documentación de procedimientos de firma. Las situaciones de evaluación realistas incluyen configuración de compilación de producción, generación de artefactos distribuidores y gestión segura de claves de firma.

Ubicación en la progresión

Este bloque se posiciona al final del ciclo de desarrollo, después de que todas las pruebas y validaciones se han completado. Los desarrolladores deben dominar estas competencias para publicar aplicación de forma segura y cumplir requisitos de tiendas de aplicaciones. El bloque combina optimizaciones técnicas con procedimientos de seguridad, preparando al equipo para publicación exitosa. La progresión natural lleva desde configuración de compilación hacia generación de artefactos finales, permitiendo que aplicación se publique de forma controlada y segura.

Bloque 8 - Seguridad, Internacionalización y Gestión del Ciclo de Vida

Objetivo pedagógico

Capacitar al desarrollador para implementar medidas de seguridad, preparar aplicación para múltiples idiomas y mercados, y gestionar ciclo de vida completo de la aplicación.

Actividades

Actividad 8.1 Aplicar principios de seguridad e identificar vulnerabilidades
Actividad 8.2 Gestionar ciclo de vida de la aplicación e implementar internacionalización
Actividad 8.3 Añadir animaciones e interacciones avanzadas

Competencias

Competencias	Indicadores

Saberes asociados

Seguridad en Aplicaciones Móviles y Vulnerabilidades OWASP

Vulnerabilidades OWASP Mobile Top 10

- Inyección de código y SQL injection
- Almacenamiento inseguro de datos
- Comunicación insegura
- Autenticación y autorización débiles
- Criptografía insuficiente
- Reverse engineering y análisis de código
- Exfiltración de datos
- Gestión insegura de sesiones

Implementación de Cifrado

- Algoritmos de cifrado simétrico (AES)
- Algoritmos de cifrado asimétrico (RSA)
- Funciones hash criptográficas (SHA-256)
- Cifrado de datos en tránsito (TLS/SSL)
- Cifrado de datos en reposo
- Gestión de claves criptográficas

Almacenamiento Seguro de Datos

- Keystore de Android
- Keychain de iOS
- Encriptación de bases de datos locales
- Protección de preferencias compartidas
- Eliminación segura de datos sensibles
- Prevención de acceso no autorizado

Revisión de Seguridad de Código

- Análisis estático de seguridad
- Herramientas de escaneo de vulnerabilidades
- Pruebas de penetración
- Auditoría de código
- Identificación de patrones inseguros
- Documentación de hallazgos de seguridad

Internacionalización y Localización de Aplicaciones Móviles

Soporte Multiidioma

- Gestión de cadenas de texto traducibles
- Archivos de recursos de idioma
- Pluralización y conjugación en diferentes idiomas
- Selección dinámica de idioma
- Fallback de idiomas
- Traducción de contenido dinámico

Adaptación Regional

- Formatos de fecha y hora según región
- Formatos de moneda y números
- Direcciones y códigos postales
- Nombres y apellidos según cultura
- Calendarios regionales
- Zonas horarias

Diseño Sensible a Cultura

- Colores y símbolos culturales
- Direccionalidad de texto (RTL/LTR)
- Imágenes y referencias culturales
- Convenciones de interacción por región
- Accesibilidad en múltiples idiomas
- Pruebas de localización

Gestión de Contenido Multiidioma

- Estrategias de carga de idiomas
- Optimización de tamaño de aplicación
- Descarga de idiomas bajo demanda
- Sincronización de traducciones
- Versionado de contenido localizado
- Mantenimiento de múltiples idiomas

Gestión del Ciclo de Vida de Aplicaciones Móviles

Planificación de Actualizaciones

- Estrategia de versionado semántico
- Planificación de ciclos de lanzamiento
- Gestión de versiones compatibles
- Deprecación de funcionalidades
- Comunicación de cambios a usuarios
- Soporte de versiones antiguas

Mantenimiento y Soporte

- Monitoreo de aplicación en producción
- Análisis de crashes y errores
- Gestión de reportes de usuarios
- Parches de seguridad

- Corrección de bugs críticos
- Documentación de cambios

Evolución de Dependencias

- Actualización de SDK de plataformas
- Actualización de librerías de terceros
- Gestión de compatibilidad hacia atrás
- Migración de APIs deprecadas
- Pruebas de compatibilidad post-actualización
- Documentación de cambios de dependencias

Fin de Vida de Aplicación

- Planificación de sunsetting
- Comunicación a usuarios
- Migración de datos de usuarios
- Eliminación de servidores backend
- Archivado de código fuente
- Documentación histórica

Mejores Prácticas de Seguridad en Desarrollo Móvil

Desarrollo Seguro

- Principio de menor privilegio
- Validación de entrada en cliente y servidor
- Sanitización de datos
- Protección contra inyección
- Gestión segura de errores
- Logging seguro sin datos sensibles

Comunicación Segura

- Uso obligatorio de HTTPS/TLS
- Validación de certificados
- Certificate pinning
- Protección contra man-in-the-middle
- Encriptación de datos en tránsito
- Autenticación mutua

Gestión de Credenciales

- Nunca hardcodear secretos
- Uso de variables de entorno
- Almacenamiento en sistemas seguros
- Rotación de claves
- Auditoría de acceso a credenciales
- Revocación de credenciales comprometidas

Pruebas de Seguridad

- Pruebas de penetración
- Análisis de seguridad estático
- Análisis de seguridad dinámico
- Fuzzing de entrada
- Pruebas de autenticación
- Pruebas de autorización

Herramientas de Monitoreo y Análisis de Aplicaciones Móviles

Monitoreo de Rendimiento

- Herramientas de profiling de CPU

- Análisis de consumo de memoria
- Monitoreo de batería
- Análisis de uso de red
- Detección de memory leaks
- Monitoreo de frame rate

Análisis de Crashes

- Plataformas de reporte de crashes
- Análisis de stack traces
- Agrupación de crashes similares
- Priorización de crashes críticos
- Reproducción de crashes
- Seguimiento de resolución

Análisis de Usuarios

- Herramientas de analytics
- Seguimiento de eventos de usuario
- Análisis de flujos de usuario
- Segmentación de usuarios
- Análisis de retención
- Análisis de conversión

Herramientas de Seguridad

- Escaneo de vulnerabilidades
- Análisis de dependencias
- Detección de código malicioso
- Monitoreo de permisos
- Auditoría de acceso a datos
- Alertas de seguridad

Postura Profesional en Seguridad e Internacionalización

Vigilancia de Seguridad

- Mentalidad de seguridad por defecto
- Actualización continua de conocimiento de amenazas
- Participación en comunidades de seguridad
- Reporte responsable de vulnerabilidades
- Documentación de decisiones de seguridad
- Comunicación clara de riesgos

Consideración de Usuarios Globales

- Empatía con usuarios de diferentes culturas
- Respeto por preferencias regionales
- Inclusión en diseño y desarrollo
- Pruebas con usuarios reales de diferentes regiones
- Apertura a feedback de diferentes mercados
- Adaptación continua a nuevos mercados

Responsabilidad a Largo Plazo

- Planificación de mantenibilidad
- Documentación para futuro
- Gestión responsable de dependencias
- Comunicación clara de cambios
- Soporte a usuarios en transiciones
- Ética en fin de vida de aplicación

Habilidades actitudinales

- Vigilancia constante de seguridad en todas las fases
- Consideración de usuarios globales en diseño
- Orientación hacia mantenibilidad a largo plazo

Justificación

Este bloque transversal agrupa competencias de seguridad y gestión que aplican a todas las fases del desarrollo. Aunque no está directamente vinculado a una función específica, es esencial para asegurar que aplicación es segura, accesible globalmente y mantenible a largo plazo. La coherencia funcional se evidencia en que todas requieren pensamiento sistémico y consideración de impacto en usuarios finales. Los entregables compatibles incluyen análisis de seguridad, configuración de internacionalización y documentación de procedimientos de mantenimiento. Las situaciones de evaluación realistas incluyen auditoría de seguridad, implementación de soporte multiidioma y planificación de actualizaciones.

Ubicación en la progresión

Este bloque se posiciona como transversal a todo el ciclo de desarrollo, siendo consideraciones que deben integrarse desde el inicio. Los desarrolladores deben mantener conciencia de seguridad, internacionalización y mantenibilidad en todas las fases del proyecto. El bloque combina consideraciones de seguridad con planificación de largo plazo, preparando al equipo para crear aplicaciones que son seguras, accesibles globalmente y fáciles de mantener. La progresión natural lleva desde implementación de medidas de seguridad básicas hacia estrategias complejas de internacionalización y mantenimiento.

Bloque 9 - Documentación, Control de Versiones e Integración Continua

Objetivo pedagógico

Capacitar al desarrollador para mantener documentación clara, gestionar código con control de versiones y automatizar procesos de integración y entrega.

Actividades

Actividad 9.1 Documentar arquitectura y participar en revisiones de código
Actividad 9.2 Integrar control de versiones y gestionar branching
Actividad 9.3 Configurar pipelines de integración y entrega continua

Competencias

Competencias	Indicadores

Saberes asociados

Sistemas de Control de Versiones

Fundamentos de Git

- Conceptos de repositorio, rama y commit
- Flujos de trabajo con Git (centralizado, feature branch, gitflow)
- Operaciones básicas: clone, add, commit, push, pull
- Resolución de conflictos de merge
- Rebase y cherry-pick

Plataformas de Alojamiento de Repositorios

- GitHub, GitLab, Bitbucket
- Gestión de permisos y acceso
- Pull requests y code review
- Integración con herramientas de CI/CD

Buenas Prácticas en Control de Versiones

- Mensajes de commit descriptivos
- Estructura de ramas (main, develop, feature, hotfix)
- Etiquetado de versiones (tagging)
- Documentación de cambios (CHANGELOG)

Integración Continua y Entrega Continua

Conceptos de CI/CD

- Definición de integración continua
- Definición de entrega continua y despliegue continuo
- Beneficios de automatización de procesos
- Pipelines de CI/CD

Herramientas de CI/CD para Desarrollo Móvil

- GitHub Actions para Android e iOS
- GitLab CI/CD
- Jenkins para proyectos móviles
- Fastlane para automatización de compilación y distribución
- Codemagic para Flutter

Configuración de Pipelines

- Definición de etapas (build, test, deploy)
- Triggers automáticos
- Gestión de secretos y credenciales
- Notificaciones de estado de pipeline
- Artefactos y reportes

Automatización de Pruebas

- Ejecución automática de pruebas unitarias
- Ejecución automática de pruebas de interfaz
- Análisis de cobertura de código
- Reportes de calidad

Documentación de Código y Arquitectura

Documentación de Código Fuente

- Comentarios en código (qué, por qué, no cómo)
- Documentación de funciones y métodos
- Documentación de clases y módulos
- Herramientas de generación de documentación (Javadoc, KDoc, Dartdoc)

Documentación de Arquitectura

- Diagramas de arquitectura (C4, UML)
- Decisiones arquitectónicas (ADR)
- Patrones de diseño utilizados
- Flujos de datos y componentes

Documentación de API

- Especificación de endpoints
- Parámetros y respuestas
- Códigos de error
- Ejemplos de uso
- Herramientas: Swagger/OpenAPI

Documentación de Procedimientos

- Guías de configuración del entorno
- Procedimientos de compilación y distribución
- Procedimientos de despliegue
- Guías de contribución

Métricas de Calidad de Código

Métricas de Cobertura

- Cobertura de líneas de código
- Cobertura de ramas
- Cobertura de funciones
- Herramientas: JaCoCo, Codecov, Coveralls

Análisis Estático de Código

- Linters y analizadores estáticos
- Detección de code smells
- Detección de vulnerabilidades

- Herramientas: SonarQube, Lint, Detekt, SwiftLint

Métricas de Complejidad

- Complejidad ciclomática
- Profundidad de anidamiento
- Tamaño de funciones y clases
- Acoplamiento y cohesión

Métricas de Mantenibilidad

- Índice de mantenibilidad
- Deuda técnica
- Duplicación de código
- Dependencias circulares

Prácticas de Colaboración en Desarrollo

Code Review

- Proceso de revisión de código
- Criterios de aceptación
- Feedback constructivo
- Herramientas de revisión (GitHub, GitLab)

Comunicación en Equipo

- Documentación clara para nuevos desarrolladores
- Explicación de decisiones técnicas
- Respuesta a preguntas sobre código
- Actualización de documentación compartida

Gestión de Cambios

- Comunicación de cambios importantes
- Coordinación de cambios entre equipos
- Gestión de dependencias entre cambios
- Planificación de despliegues

Buenas Prácticas en Automatización

Identificación de Procesos Automatizables

- Procesos repetitivos y manuales
- Procesos propensos a errores
- Procesos que consumen mucho tiempo
- Análisis de costo-beneficio de automatización

Diseño de Automatización

- Definición de criterios de éxito
- Manejo de casos excepcionales
- Monitoreo y alertas
- Documentación de automatización

Mejora Continua

- Monitoreo de métricas de automatización
- Identificación de cuellos de botella
- Optimización de pipelines
- Actualización de procesos automatizados

Habilidades actitudinales

- Disciplina en documentación de código y decisiones
- Rigor en control de versiones

- Proactividad en automatización de procesos
- Colaboración efectiva a través de documentación clara

Justificación

Este bloque transversal agrupa competencias de infraestructura técnica y documentación que soportan todo el ciclo de desarrollo. Aunque no está directamente vinculado a una función específica, es esencial para colaboración efectiva del equipo y automatización de procesos. La coherencia funcional se evidencia en que todas requieren disciplina, atención al detalle y pensamiento sistémico. Los entregables compatibles incluyen documentación técnica, repositorio de código bien organizado, pipelines de CI/CD y reportes de calidad. Las situaciones de evaluación realistas incluyen creación de documentación de API, gestión de ramas de desarrollo, configuración de pipelines de testing automático y análisis de métricas de calidad.

Ubicación en la progresión

Este bloque se posiciona como transversal a todo el ciclo de desarrollo, siendo prácticas que deben establecerse desde el inicio del proyecto. Los desarrolladores deben mantener disciplina en documentación y control de versiones en todas las fases del proyecto. El bloque combina herramientas técnicas con prácticas de colaboración, preparando al equipo para trabajar de forma eficiente y escalable. La progresión natural lleva desde control de versiones básico hacia pipelines complejos de integración continua, permitiendo que equipo automatice procesos y mantenga calidad consistente.

ANEXO IV Marco de evaluación

ANEXO IV a. Exenciones de unidades

ANEXO IV b. Reglamento de examen

Pruebas	Unidad	Coef.	Forma	Duración
E1 Análisis y Especificación de Requisitos de Aplicaciones Móviles	**U1**	**2**	Prueba escrita única	3h
E2 Selección de Plataforma y Configuración del Entorno de Desarrollo	**U2**	**2**	Prueba práctica única	4h
E3 Diseño de Arquitectura e Interfaz de Aplicaciones Móviles	**U3**	**3**	Prueba escrita única	4h
E4 Desarrollo de Funcionalidades y Lógica de Negocio	**U4**	**3**	Prueba práctica única	5h
E5 Integración de Servicios Avanzados y Funcionalidades Especializadas	**U5**	**2**	Prueba práctica única	4h
E6 Pruebas, Depuración y Optimización de Aplicaciones Móviles	**U6**	**2**	Prueba práctica única	4h
E7 Publicación, Actualización y Mantenimiento de Aplicaciones Móviles	**U7**	**1**	Prueba práctica única	2h
E8 Seguridad, Internacionalización y Gestión del Ciclo de Vida	**U8**	**1**	Prueba escrita única	2h
E9 Documentación, Control de Versiones e Integración Continua	**U9**	**1**	Prueba escrita única	2h

*Pruebas optativas: solo se tienen en cuenta los puntos por encima de la media.

ANEXO IV c. Definición de las pruebas

E1 : Análisis y Especificación de Requisitos de Aplicaciones Móviles

Finalidades de la prueba

Evaluar capacidad del candidato para recopilar, analizar y especificar requisitos funcionales y no funcionales de aplicaciones móviles, asegurando alineación con objetivos de negocio y restricciones técnicas.

Contenido de la prueba

Recopilación de requisitos funcionales mediante entrevistas y talleres; documentación de historias de usuario y casos de uso; especificación de requisitos no funcionales de rendimiento, seguridad y compatibilidad; definición y negociación de alcance del producto; priorización de funcionalidades según valor de negocio y esfuerzo técnico; traducción de necesidades de negocio en especificaciones técnicas; gestión de cambios de requisitos.

Competencias evaluadas

C1 - Recopilar y documentar requisitos funcionales de la aplicación móvil con stakeholders para asegurar la alineación con objetivos de negocio

C2 - Analizar y especificar requisitos no funcionales de rendimiento, seguridad y compatibilidad para definir restricciones técnicas móviles

C3 - Definir y negociar el alcance del producto móvil con stakeholders para establecer límites claros y gestionar cambios

C4 - Priorizar historias de usuario y funcionalidades según valor de negocio y esfuerzo técnico para optimizar entregas incrementales

C5 - Traducir necesidades de negocio en especificaciones técnicas funcionales para facilitar comunicación entre equipos de negocio y desarrollo

Criterios de evaluación

- Documenta requisitos funcionales con trazabilidad clara a fuentes y criterios de aceptación verificables
- Especifica requisitos no funcionales con métricas cuantificables y restricciones técnicas móviles identificadas
- Define alcance del producto con límites claros entre funcionalidades incluidas y excluidas
- Prioriza funcionalidades utilizando técnicas estructuradas considerando valor de negocio y esfuerzo técnico
- Traduce necesidades de negocio en especificaciones técnicas comprensibles para equipo de desarrollo
- Mantiene matrices de trazabilidad actualizadas y documentación de decisiones con justificación clara
- Valida requisitos con stakeholders técnicos y de negocio obteniendo aprobación documentada

Forma de evaluación

Vía escolar pública o privada concertada : Prueba escrita única — Análisis de caso de negocio real con recopilación de requisitos, especificación de funcionalidades y documentación de alcance

Aprendizaje : Evaluación continua — Evaluación en contexto de trabajo real mediante análisis de requisitos en proyectos de empresa

Formación continua : Evaluación continua — Evaluación mediante análisis de requisitos en proyectos reales de formación continua

Vae : Prueba escrita única — Análisis de portafolio de proyectos anteriores y prueba de análisis de requisitos

Reglas específicas

- Candidato debe demostrar capacidad de comunicación efectiva con stakeholders no técnicos
- Documentación debe incluir trazabilidad clara entre requisitos de negocio y especificaciones técnicas
- Requisitos especificados deben ser verificables y medibles
- Alcance debe estar claramente delimitado con justificación de decisiones

E2 : Selección de Plataforma y Configuración del Entorno de Desarrollo

Finalidades de la prueba

Evaluar capacidad del candidato para evaluar ecosistemas móviles, seleccionar plataformas de desarrollo, configurar entornos de trabajo y gestionar dependencias técnicas.

Contenido de la prueba

Evaluación de ecosistemas Android e iOS; análisis de restricciones de dispositivos y versiones de sistema operativo; comparación de enfoques nativo vs multiplataforma; evaluación de frameworks multiplataforma; instalación y configuración de IDEs (Android Studio, Xcode); configuración de emuladores y dispositivos virtuales; conexión con dispositivos físicos; gestión de dependencias con Gradle y CocoaPods; resolución de conflictos de compatibilidad.

Competencias evaluadas

C7 - Evaluar y seleccionar plataforma de desarrollo (Android, iOS o multiplataforma) basándose en requisitos técnicos y restricciones de negocio

C8 - Configurar entorno de desarrollo móvil instalando IDEs, emuladores y dependencias para preparar infraestructura de codificación

C9 - Gestionar dependencias y SDK del proyecto móvil resolviendo conflictos de compatibilidad para mantener estabilidad de compilación

Criterios de evaluación

- Realiza análisis comparativo de plataformas considerando requisitos técnicos, cuota de mercado y restricciones de negocio
- Documenta decisión de plataforma con justificación clara de criterios de evaluación y trade-offs identificados
- Configura entorno de desarrollo completo incluyendo IDE, emuladores y dispositivos físicos
- Valida que todas las herramientas están correctamente instaladas y funcionan sin errores
- Gestiona dependencias resolviendo conflictos de compatibilidad de forma sistemática
- Documenta procedimientos de configuración para referencia futura y onboarding de nuevos desarrolladores
- Identifica y resuelve problemas de compatibilidad de dependencias antes de que causen fallos

Forma de evaluación

Vía escolar pública o privada concertada : Prueba práctica única — Configuración completa de entorno de desarrollo para proyecto móvil con resolución de conflictos de dependencias

Aprendizaje : Evaluación continua — Evaluación en contexto de trabajo mediante configuración de entornos en proyectos de empresa

Formación continua : Evaluación continua — Evaluación mediante configuración de entornos en proyectos reales de formación continua

Vae : Prueba práctica única — Configuración de entorno y resolución de problemas técnicos en contexto de experiencia previa

Reglas específicas

- Candidato debe documentar todas las decisiones técnicas con justificación clara
- Entorno configurado debe estar completamente funcional sin errores de compilación
- Todas las dependencias deben estar correctamente versionadas y documentadas
- Procedimientos de configuración deben ser reproducibles por otros desarrolladores

E3 : Diseño de Arquitectura e Interfaz de Aplicaciones Móviles

Finalidades de la prueba

Evaluar capacidad del candidato para diseñar arquitectura de información, interfaces de usuario escalables y patrones de navegación centrados en usuario.

Contenido de la prueba

Diseño de arquitectura de información y modelado de datos; diseño de flujos de navegación; aplicación de patrones arquitectónicos (MVC, MVP, MVVM); diseño de interfaces siguiendo Material Design y Human Interface Guidelines; diseño de layouts responsivos y adaptativos; creación de componentes reutilizables; gestión de estado de aplicación; diseño centrado en usuario con prototipos de baja y alta fidelidad; validación de diseños con stakeholders; implementación de accesibilidad.

Competencias evaluadas

C6 - Diseñar arquitectura de información y modelado de datos para estructurar la aplicación móvil de forma escalable y mantenible

C10 - Implementar lógica de negocio aplicando patrones de arquitectura (MVC, MVP, MVVM) para asegurar separación de responsabilidades y mantenibilidad

C11 - Implementar navegación entre pantallas de la aplicación móvil gestionando flujos de usuario y paso de parámetros para crear experiencia coherente

C12 - Desarrollar interfaces de usuario móviles siguiendo directrices de diseño (Material Design, HIG) para asegurar consistencia y accesibilidad

C13 - Construir layouts adaptativos y responsivos para múltiples tamaños de pantalla utilizando unidades relativas y restricciones de layout

C14 - Crear componentes reutilizables de interfaz documentando API y manteniendo biblioteca centralizada para optimizar desarrollo y consistencia

C15 - Gestionar estado de la aplicación móvil implementando patrones de gestión (Redux, BLoC, Provider) para sincronizar datos entre componentes

C16 - Integrar patrones de diseño centrado en el usuario recopilando feedback y validando interacciones para iterar basándose en necesidades reales

C17 - Diseñar prototipos de pantallas móviles de baja y alta fidelidad utilizando herramientas de diseño para validar conceptos antes de desarrollo

C18 - Validar wireframes y prototipos con stakeholders obteniendo aprobación formal antes de iniciar desarrollo de funcionalidades

Criterios de evaluación

• Diseña arquitectura de información clara con modelado de datos escalable y documentado
• Aplica patrones arquitectónicos de forma consistente en toda la aplicación
• Diseña flujos de navegación intuitivos con transiciones claras entre pantallas
• Implementa interfaces siguiendo directrices de diseño (Material Design, HIG) con consistencia visual
• Crea layouts responsivos que se adaptan a múltiples tamaños de pantalla sin pérdida de usabilidad
• Identifica y abstrae componentes reutilizables documentando API y manteniendo biblioteca centralizada
• Implementa gestión de estado eficiente utilizando patrones apropiados (Redux, BLoC, Provider)
• Valida diseños con usuarios reales mediante pruebas de usabilidad e itera basándose en feedback
• Crea prototipos de baja y alta fidelidad para validar conceptos antes de desarrollo
• Obtiene aprobación formal de stakeholders antes de iniciar desarrollo de funcionalidades
• Implementa características de accesibilidad para usuarios con discapacidades

Forma de evaluación

Vía escolar pública o privada concertada : Prueba escrita única — Diseño completo de arquitectura e interfaz de aplicación móvil con documentación de decisiones

Vía escolar pública o privada concertada : Prueba práctica única — Implementación de componentes visuales y validación de accesibilidad

Aprendizaje : Evaluación continua — Evaluación en contexto de trabajo mediante diseño de interfaces en proyectos de empresa

Formación continua : Evaluación continua — Evaluación mediante diseño de arquitectura en proyectos reales de formación continua

Vae : Prueba escrita única — Análisis de portafolio de diseños anteriores y diseño de nueva interfaz

Reglas específicas

• Arquitectura debe ser escalable y permitir extensión futura sin cambios mayores
• Interfaces deben cumplir directrices de accesibilidad WCAG 2.1 nivel AA
• Componentes reutilizables deben estar documentados con ejemplos de uso
• Diseños deben ser validados con usuarios reales antes de aprobación final
• Patrones arquitectónicos deben aplicarse de forma consistente en toda la aplicación

E4 : Desarrollo de Funcionalidades y Lógica de Negocio

Finalidades de la prueba

Evaluar capacidad del candidato para implementar lógica de negocio, persistencia de datos y comunicación con servicios externos de forma segura y mantenible.

Contenido de la prueba

Implementación de persistencia local utilizando SQLite, Room (Android) y Core Data (iOS); conexión con APIs externas mediante clientes HTTP; consumo de servicios REST; procesamiento de respuestas JSON; validación de datos de usuario; implementación de formularios con validación cliente-servidor; gestión de errores y excepciones; manejo de fallos de conectividad; implementación de repositorios para abstracción de acceso a datos.

Competencias evaluadas

C19 - Implementar persistencia local de datos utilizando bases de datos móviles (SQLite, Room, Core Data) con gestión de migraciones

C20 - Conectar aplicación móvil con APIs externas implementando clientes HTTP y gestionando errores de red para asegurar comunicación confiable

C21 - Consumir servicios web REST desde aplicación móvil implementando métodos HTTP y abstrayendo lógica de red en repositorios

C22 - Procesar respuestas JSON de servicios remotos deserializando datos y mapeándolos a modelos de aplicación con validación

C27 - Desarrollar formularios de entrada de datos con validación, gestión de teclado virtual y navegación accesible entre campos

C28 - Validar campos y entradas del usuario implementando validaciones cliente y servidor con mensajes de error contextuales

Criterios de evaluación

- Implementa persistencia local con esquema de datos bien diseñado y migraciones controladas
- Conecta aplicación con APIs externas implementando clientes HTTP robustos
- Consume servicios REST implementando métodos HTTP correctamente y gestionando códigos de respuesta
- Procesa respuestas JSON deserializando datos y mapeándolos a modelos con validación
- Implementa validaciones de entrada en cliente y servidor con mensajes de error contextuales
- Desarrolla formularios con validación, gestión de teclado virtual y navegación accesible
- Gestiona errores y excepciones implementando manejo robusto con logging y análisis
- Implementa repositorios que abstraen lógica de acceso a datos (local y remoto)
- Maneja fallos de conectividad permitiendo sincronización cuando red se restaura
- Valida integridad de datos en base de datos local y transacciones multi-paso

Forma de evaluación

Vía escolar pública o privada concertada : Prueba práctica única — Implementación de funcionalidades completas incluyendo persistencia, APIs y validación

Aprendizaje : Evaluación continua — Evaluación en contexto de trabajo mediante implementación de funcionalidades en proyectos de empresa

Formación continua : Evaluación continua — Evaluación mediante implementación de funcionalidades en proyectos reales de formación continua

Vae : Prueba práctica única — Implementación de funcionalidades complejas demostrando dominio de persistencia y APIs

Reglas específicas

- Código debe seguir patrones de arquitectura definidos en diseño
- Validaciones deben implementarse tanto en cliente como en servidor
- Manejo de errores debe ser robusto y permitir recuperación de fallos
- Acceso a datos debe estar abstraído en repositorios reutilizables
- Datos sensibles deben ser protegidos en tránsito y en reposo

E5 : Integración de Servicios Avanzados y Funcionalidades Especializadas

Finalidades de la prueba

Evaluar capacidad del candidato para integrar servicios avanzados como autenticación, notificaciones push y seguridad, implementando funcionalidades especializadas.

Contenido de la prueba

Implementación de flujos de autenticación (OAuth, Firebase Auth); gestión segura de tokens y sesiones; integración de notificaciones push (Firebase Cloud Messaging, Apple Push Notification); implementación de animaciones e interacciones; análisis de seguridad y vulnerabilidades OWASP Mobile; implementación de cifrado y almacenamiento seguro de datos; revisión de seguridad de código.

Competencias evaluadas

C23 - Integrar autenticación de usuarios implementando flujos OAuth, Firebase Auth o autenticación personalizada con recuperación de contraseña

C24 - Gestionar sesiones y tokens de acceso almacenando de forma segura y renovando automáticamente para mantener autenticación persistente

C25 - Implementar notificaciones push integrando Firebase Cloud Messaging o Apple Push Notification service con gestión de permisos

C26 - Añadir animaciones e interacciones en interfaz móvil utilizando APIs nativas optimizando rendimiento para mantener fluidez

C34 - Aplicar principios de seguridad en desarrollo identificando vulnerabilidades OWASP Mobile e implementando cifrado y almacenamiento seguro

Criterios de evaluación

- Implementa flujos de autenticación seguros con recuperación de contraseña y manejo de errores
- Integra OAuth o Firebase Auth permitiendo autenticación mediante proveedores externos
- Gestiona tokens de acceso almacenándolos de forma segura y renovándolos automáticamente
- Implementa notificaciones push con gestión de permisos y manejo en primer y segundo plano
- Añade animaciones e interacciones que mejoran experiencia sin impactar rendimiento
- Identifica vulnerabilidades OWASP Mobile en código y especifica mitigaciones
- Implementa cifrado para datos sensibles en tránsito y en reposo
- Realiza revisión de seguridad de código identificando prácticas inseguras
- Almacena datos sensibles en sistemas seguros del dispositivo (Keystore, Keychain)
- Documenta medidas de seguridad implementadas y justificación de decisiones

Forma de evaluación

Vía escolar pública o privada concertada : Prueba práctica única — Implementación de autenticación, notificaciones y análisis de seguridad

Aprendizaje : Evaluación continua — Evaluación en contexto de trabajo mediante integración de servicios avanzados

Formación continua : Evaluación continua — Evaluación mediante integración de servicios en proyectos reales

Vae : Prueba práctica única — Implementación de servicios avanzados demostrando dominio de seguridad

Reglas específicas

- Autenticación debe ser segura con protección contra ataques comunes
- Tokens deben almacenarse en sistemas seguros del dispositivo
- Análisis de seguridad debe identificar vulnerabilidades OWASP Mobile
- Datos sensibles deben ser cifrados en tránsito y en reposo
- Notificaciones deben respetar permisos y preferencias del usuario

E6 : Pruebas, Depuración y Optimización de Aplicaciones Móviles

Finalidades de la prueba

Evaluar capacidad del candidato para escribir pruebas automatizadas, depurar errores complejos y validar compatibilidad en múltiples dispositivos.

Contenido de la prueba

Escritura de pruebas unitarias con JUnit, XCTest, Flutter Test; desarrollo de pruebas automatizadas de interfaz con Espresso, XCUITest, Appium; depuración de fallos utilizando herramientas de debugging; análisis de logs y volcados de memoria; ejecución de pruebas de compatibilidad en matriz de dispositivos; documentación de problemas de compatibilidad; priorización de defectos; gestión de errores y excepciones.

Competencias evaluadas

C29 - Gestionar errores y excepciones de ejecución implementando manejo robusto con logging y análisis para mejorar confiabilidad

C30 - Depurar fallos y errores de aplicación móvil utilizando herramientas de depuración, análisis de logs y reproducción de errores

C31 - Escribir pruebas unitarias para lógica de negocio utilizando frameworks de testing (JUnit, XCTest, Flutter Test) con cobertura adecuada

C32 - Desarrollar pruebas automatizadas de interfaz de usuario implementando escenarios de prueba con herramientas de testing (Espresso, XCUITest, Appium)

C33 - Ejecutar pruebas de compatibilidad en matriz de dispositivos documentando problemas y priorizando defectos para asegurar funcionamiento multiplataforma

Criterios de evaluación

- Escribe pruebas unitarias con cobertura adecuada de lógica de negocio
- Desarrolla pruebas automatizadas de interfaz que validan flujos de usuario críticos
- Utiliza herramientas de debugging para identificar y resolver errores complejos
- Analiza logs y trazas de error para diagnosticar problemas de forma sistemática
- Ejecuta pruebas de compatibilidad en matriz de dispositivos y versiones de SO
- Documenta problemas de compatibilidad con pasos de reproducción claros
- Prioriza defectos según severidad e impacto en usuarios
- Implementa manejo robusto de errores con logging y análisis
- Valida que soluciones resuelven problemas sin crear otros
- Mantiene pruebas actualizadas con cambios de código

Forma de evaluación

Vía escolar pública o privada concertada : Prueba práctica única — Escritura de pruebas unitarias y automatizadas, depuración de errores y pruebas de compatibilidad

Aprendizaje : Evaluación continua — Evaluación en contexto de trabajo mediante testing y depuración en proyectos de empresa

Formación continua : Evaluación continua — Evaluación mediante testing en proyectos reales de formación continua

Vae : Prueba práctica única — Demostración de testing y depuración en proyectos anteriores

Reglas específicas

- Pruebas unitarias deben tener cobertura mínima del 70% de código crítico
- Pruebas automatizadas deben validar flujos de usuario completos
- Defectos deben documentarse con pasos de reproducción claros
- Compatibilidad debe validarse en al menos 3 dispositivos diferentes
- Errores deben ser analizados sistemáticamente antes de implementar soluciones

E7 : Publicación, Actualización y Mantenimiento de Aplicaciones Móviles

Finalidades de la prueba

Evaluar capacidad del candidato para preparar compilaciones de producción, gestionar claves de firma y publicar aplicaciones en tiendas de aplicaciones.

Contenido de la prueba

Configuración de compilaciones de producción; optimización de artefactos; generación de APK y AAB para Android; generación de IPA para iOS; gestión segura de claves de firma; cumplimiento de requisitos de tiendas de aplicaciones; documentación de procedimientos de publicación.

Competencias evaluadas

C35 - Preparar compilaciones para publicación configurando parámetros de producción, generando artefactos y gestionando claves de firma

Criterios de evaluación

- Configura compilaciones de producción con todas las optimizaciones habilitadas
- Genera artefactos distribuidores (APK, AAB, IPA) correctamente
- Gestiona claves de firma de forma segura en sistemas protegidos
- Verifica que aplicación cumple requisitos de Google Play y App Store
- Documenta procedimientos de publicación para referencia futura
- Valida artefactos antes de publicación
- Obtiene aprobación de cambios antes de publicación
- Implementa versionado correcto de aplicación

Forma de evaluación

Vía escolar pública o privada concertada : Prueba práctica única — Configuración de compilación de producción y generación de artefactos distribuidores

Aprendizaje : Evaluación continua — Evaluación en contexto de trabajo mediante preparación de compilaciones

Formación continua : Evaluación continua — Evaluación mediante preparación de compilaciones en proyectos reales

Vae : Prueba práctica única — Demostración de preparación de compilaciones en proyectos anteriores

Reglas específicas

- Claves de firma deben almacenarse en sistemas seguros
- Compilaciones deben estar optimizadas para producción
- Artefactos deben ser validados antes de publicación

- Procedimientos deben estar documentados para reproducibilidad
- Cumplimiento de requisitos de tiendas debe ser verificado antes de publicación

E8 : Seguridad, Internacionalización y Gestión del Ciclo de Vida

Finalidades de la prueba

Evaluar capacidad del candidato para implementar medidas de seguridad transversales, preparar aplicación para múltiples idiomas y mercados, y gestionar ciclo de vida.

Contenido de la prueba

Análisis de seguridad en todas las fases; implementación de medidas de seguridad recomendadas; internacionalización de interfaz y contenido; adaptación a diferentes regiones y idiomas; gestión de ciclo de vida de aplicación; planificación de actualizaciones; mantenimiento a largo plazo.

Competencias evaluadas

Criterios de evaluación

- Implementa medidas de seguridad en todas las fases del desarrollo
- Mantiene conocimiento actualizado de vulnerabilidades y amenazas
- Diseña interfaz considerando múltiples idiomas y longitudes de texto
- Valida que textos se adaptan correctamente a diferentes idiomas
- Planifica actualizaciones de dependencias de forma sistemática
- Documenta decisiones técnicas para referencia futura
- Implementa estrategias de mantenimiento a largo plazo

Forma de evaluación

Vía escolar pública o privada concertada : Prueba escrita única — Análisis de seguridad y planificación de internacionalización

Aprendizaje : Evaluación continua — Evaluación en contexto de trabajo mediante análisis de seguridad

Formación continua : Evaluación continua — Evaluación mediante análisis de seguridad en proyectos reales

Vae : Prueba escrita única — Análisis de seguridad y planificación de mantenimiento

Reglas específicas

- Análisis de seguridad debe identificar vulnerabilidades OWASP Mobile
- Internacionalización debe soportar al menos 3 idiomas
- Medidas de seguridad deben estar documentadas y justificadas
- Planificación de mantenimiento debe cubrir al menos 2 años

E9 : Documentación, Control de Versiones e Integración Continua

Finalidades de la prueba

Evaluar capacidad del candidato para mantener documentación clara, gestionar código con control de versiones y automatizar procesos de integración.

Contenido de la prueba

Documentación de código y decisiones técnicas; control de versiones con Git; gestión de ramas de desarrollo; escritura de commits descriptivos; configuración de pipelines de CI/CD; automatización de pruebas; análisis de métricas de calidad; colaboración efectiva mediante documentación.

Competencias evaluadas

Criterios de evaluación

- Documenta propósito de funciones complejas y decisiones de diseño no obvias
- Realiza commits con mensajes descriptivos agrupando cambios relacionados
- Revisa cambios antes de hacer push a repositorio
- Mantiene documentación actualizada con cambios de código
- Configura pipelines de CI/CD para ejecutar pruebas automáticamente
- Monitorea métricas de calidad de código
- Escribe documentación clara para desarrolladores nuevos
- Mantiene wiki o documentación centralizada actualizada

Forma de evaluación

Vía escolar pública o privada concertada : Prueba escrita única — Documentación de código y configuración de control de versiones

Aprendizaje : Evaluación continua — Evaluación en contexto de trabajo mediante documentación y control de versiones

Formación continua : Evaluación continua — Evaluación mediante documentación en proyectos reales

Vae : Prueba escrita única — Análisis de documentación y control de versiones en proyectos anteriores

Reglas específicas

- Documentación debe ser clara y accesible para desarrolladores nuevos
- Commits deben tener mensajes descriptivos y cambios relacionados
- Pipelines de CI/CD deben ejecutar pruebas automáticamente
- Métricas de calidad deben ser monitoreadas regularmente
- Documentación debe estar centralizada y actualizada

ANEXO IV d. Dispositivo de evaluación detallado por bloque

Este anexo describe, para cada bloque de competencias, el dispositivo de evaluación detallado: indicadores SMART por misión profesional, situaciones de evaluación (simulaciones, estudios de caso), pruebas esperadas (producciones documentales, acciones observables, elementos reflexivos), criterios de éxito agrupados en siete familias, y el umbral de validación del bloque. Este dispositivo constituye la referencia para evaluadores y tribunales.

Bloque 1 - Análisis y Especificación de Requisitos de Aplicaciones Móviles

Indicadores de desempeño por misión

Misión	Indicadores SMART
M05 — Analizar requisitos funcionales de la aplicación móvil	Número de requisitos funcionales documentados con trazabilidad clara a fuente Porcentaje de requisitos validados con stakeholders antes de finalizar Número de casos de uso documentados con flujos normales y excepcionales Porcentaje de historias de usuario con criterios de aceptación completos
M06 — Analizar requisitos no funcionales de la aplicación móvil	Número de requisitos no funcionales identificados y documentados Porcentaje de restricciones técnicas móviles evaluadas Número de métricas de rendimiento definidas con valores objetivo Porcentaje de criterios de aceptación especificados para requisitos no funcionales
M07 — Definir el alcance funcional del producto móvil	Número de funcionalidades incluidas en alcance documentadas Número de funcionalidades excluidas en alcance documentadas Porcentaje de cambios de alcance evaluados y documentados Número de aprobaciones de stakeholders obtenidas para alcance
M08 — Priorizar historias de usuario para la entrega móvil	Número de historias de usuario priorizadas en backlog Porcentaje de historias con evaluación de valor de negocio Porcentaje de historias con estimación de esfuerzo técnico Número de dependencias identificadas y documentadas
M09 — Traducir necesidades de negocio en funcionalidades móviles	Número de requisitos de negocio traducidos a especificaciones técnicas Porcentaje de especificaciones técnicas validadas con equipo de desarrollo Número de sesiones de facilitación de comunicación realizadas Porcentaje de requisitos con trazabilidad documentada

Situaciones de evaluación

Sesión de Recopilación de Requisitos con Cliente

Contexto : El desarrollador participa en sesión de recopilación de requisitos con cliente de aplicación móvil de comercio electrónico. El cliente describe necesidades de negocio de forma general y el desarrollador debe identificar requisitos funcionales específicos, documentar casos de uso y redactar historias de usuario.

Restricciones : Sesión limitada a 2 horas, cliente disponible solo en horarios específicos, múltiples stakeholders con perspectivas diferentes

Producciones esperadas :

- Notas de sesión con requisitos identificados
- Casos de uso documentados con flujos normales y excepcionales
- Historias de usuario redactadas con criterios de aceptación
- Lista de preguntas de clarificación para seguimiento

Competencias evaluadas : C1, C5

Análisis de Documentación de Negocio y Especificación de Requisitos No Funcionales

Contexto : El desarrollador recibe documentación de negocio de aplicación móvil de banca digital y debe analizar requisitos no funcionales de rendimiento, seguridad y compatibilidad. Debe identificar restricciones técnicas móviles y definir métricas de rendimiento.

Restricciones : Documentación incompleta, requisitos no funcionales implícitos, múltiples plataformas objetivo (Android e iOS)

Producciones esperadas :

- Matriz de requisitos no funcionales identificados
- Evaluación de restricciones técnicas móviles
- Definición de métricas de rendimiento con valores objetivo
- Especificación de criterios de aceptación para requisitos no funcionales

Competencias evaluadas : C2, C5

Definición y Negociación de Alcance con Stakeholders

Contexto : El desarrollador trabaja con equipo de producto y cliente para definir alcance de aplicación móvil de redes sociales. Debe documentar funcionalidades incluidas y excluidas, negociar requisitos conflictivos y gestionar cambios de alcance.

Restricciones : Presupuesto limitado, cronograma ajustado, múltiples stakeholders con prioridades conflictivas

Producciones esperadas :

- Documento de alcance del producto
- Matriz de trazabilidad de requisitos
- Documentación de negociaciones y acuerdos
- Plan de gestión de cambios de alcance

Competencias evaluadas : C3, C4

Priorización de Backlog y Estimación de Esfuerzo

Contexto : El desarrollador participa en sesión de priorización de backlog para aplicación móvil de productividad. Debe evaluar valor de negocio de cada historia de usuario, estimar esfuerzo técnico e identificar dependencias.

Restricciones : Múltiples criterios de priorización, estimaciones inciertas, dependencias técnicas complejas

Producciones esperadas :

- Backlog priorizado con justificación
- Estimaciones de esfuerzo técnico documentadas
- Matriz de dependencias entre historias
- Plan de entregas incrementales

Competencias evaluadas : C4, C5

Validación de Especificaciones Técnicas con Equipo de Desarrollo

Contexto : El desarrollador presenta especificaciones técnicas de aplicación móvil de salud a equipo de desarrollo. Debe traducir requisitos de negocio en especificaciones técnicas claras, facilitar comunicación entre equipos y asegurar trazabilidad de requisitos.

Restricciones : Equipo técnico con diferentes niveles de experiencia, requisitos complejos, necesidad de validación rápida

Producciones esperadas :

- Especificaciones técnicas detalladas
- Diagramas de arquitectura de información
- Matriz de trazabilidad requisitos-especificaciones
- Acta de validación con equipo de desarrollo

Competencias evaluadas : C1, C2, C5

Pruebas esperadas

■ Pruebas documentales

- Documento de especificación de requisitos funcionales
- Documento de especificación de requisitos no funcionales
- Casos de uso documentados
- Historias de usuario redactadas
- Matriz de trazabilidad de requisitos
- Documento de alcance del producto
- Backlog priorizado
- Especificaciones técnicas
- Actas de validación con stakeholders
- Documentación de decisiones técnicas

■ Pruebas de acción (en campo)

- Realización de entrevistas con stakeholders
- Facilitación de talleres de requisitos
- Documentación de casos de uso
- Redacción de historias de usuario
- Validación de requisitos con stakeholders
- Negociación de alcance
- Priorización de backlog

- Estimación de esfuerzo técnico
- Identificación de dependencias
- Presentación de especificaciones técnicas

■ **Pruebas reflexivas**

- Análisis de completitud de requisitos documentados
- Evaluación de claridad de especificaciones técnicas
- Reflexión sobre efectividad de comunicación con stakeholders
- Análisis de precisión de estimaciones de esfuerzo
- Evaluación de calidad de trazabilidad de requisitos
- Reflexión sobre gestión de conflictos en negociación
- Análisis de impacto de cambios de alcance
- Evaluación de alineación entre requisitos de negocio y técnicos

Criterios de éxito (7 familias)

Familia	Definición	Ponderación
Conformité	Todos los requisitos funcionales están documentados con trazabilidad clara a fuente y validados con stakeholders	100% de requisitos con trazabilidad documentada y validación de stakeholders
Qualité	Especificaciones técnicas son claras, completas y sin ambigüedades, permitiendo que equipo de desarrollo entienda exactamente qué implementar	Equipo de desarrollo valida que especificaciones son suficientemente detalladas para iniciar desarrollo
Comunicación	Comunicación con stakeholders es efectiva, clara y bidireccional, asegurando comprensión mutua de requisitos	Stakeholders confirman que requisitos documentados reflejan exactamente sus necesidades
Pertinence	Requisitos identificados son relevantes para objetivos de negocio y necesidades de usuarios finales	Cada requisito puede ser vinculado a objetivo de negocio o necesidad de usuario documentada
Cohérence	Requisitos funcionales y no funcionales son consistentes entre sí sin conflictos o contradicciones	Análisis de matriz de requisitos no identifica conflictos o contradicciones
Traçabilité	Existe trazabilidad clara desde requisitos de negocio hasta especificaciones técnicas y criterios de aceptación	Matriz de trazabilidad completa con vinculación bidireccional entre elementos
Posture	Desarrollador demuestra escucha activa, rigor en documentación, análisis objetivo y negociación constructiva con stakeholders	Stakeholders y equipo técnico validan que desarrollador demostró actitudes profesionales esperadas

Umbral de validación : Promedio ≥ 10/20

Bloque 2 - Selección de Plataforma y Configuración del Entorno de Desarrollo

Indicadores de desempeño por misión

Misión	Indicadores SMART
M11 — Seleccionar la plataforma objetivo Android para el desarrollo	Matriz de evaluación de plataforma Android completada con al menos 5 criterios técnicos y de negocio Documentación de decisión de selección de Android incluyendo análisis de cuota de mercado y restricciones de dispositivos Justificación técnica de selección con referencias a requisitos funcionales y no funcionales del proyecto Validación de decisión con stakeholders técnicos y de negocio documentada
M12 — Seleccionar la plataforma objetivo iOS para el desarrollo	Matriz de evaluación de plataforma iOS completada con al menos 5 criterios técnicos y de negocio Documentación de decisión de selección de iOS incluyendo análisis de directrices de Apple y restricciones de dispositivos Justificación técnica de selección con referencias a requisitos funcionales y no funcionales del proyecto Validación de decisión con stakeholders técnicos y de negocio documentada
M13 — Elegir un enfoque nativo para el desarrollo móvil	Análisis comparativo de desarrollo nativo vs multiplataforma completado con al menos 6 criterios de evaluación Documentación de impacto en rendimiento, mantenibilidad y costo de cada enfoque Justificación de selección de enfoque nativo con referencias a requisitos del proyecto Validación de decisión con equipo técnico documentada
M14 — Elegir un enfoque multiplataforma para el desarrollo móvil	Evaluación de frameworks multiplataforma (React Native, Flutter, Xamarin) completada con al menos 6 criterios Análisis de impacto en rendimiento, mantenibilidad y adopción de cada framework Documentación de configuración de proyecto base con framework seleccionado Validación de decisión con equipo técnico y prueba de compilación exitosa
M15 — Configurar el entorno de desarrollo móvil	Entorno de desarrollo completamente configurado con IDE, SDK y herramientas de compilación instalados Emuladores y dispositivos virtuales creados y validados para múltiples versiones de SO Dispositivos físicos conectados y validados para depuración Proyecto base compilado y ejecutado exitosamente en emulador, simulador y dispositivo físico

Situaciones de evaluación

Evaluación de plataforma Android para nuevo proyecto

Contexto : Se inicia un nuevo proyecto de aplicación móvil para una empresa de comercio electrónico. El equipo debe evaluar si Android es la plataforma objetivo apropiada considerando requisitos técnicos (rendimiento, seguridad, integraciones), restricciones de negocio (presupuesto, cronograma) y análisis de mercado (cuota de usuarios, dispositivos objetivo).

Restricciones : Debe completarse en máximo 2 semanas. Debe incluir análisis de al menos 3 alternativas de plataforma. Debe validarse con stakeholders de negocio y técnicos.

Producciones esperadas :

- Matriz de evaluación de plataformas con criterios ponderados
- Análisis de cuota de mercado Android en región objetivo
- Documento de restricciones de dispositivos Android identificadas
- Recomendación formal de selección de plataforma con justificación

Competencias evaluadas : C7

Configuración de entorno de desarrollo Android desde cero

Contexto : Un nuevo desarrollador se incorpora al equipo y debe configurar su entorno de desarrollo para trabajar en proyecto Android existente. Debe instalar Android Studio, configurar SDK, crear emuladores, conectar dispositivos físicos y validar que puede compilar y ejecutar la aplicación.

Restricciones : Debe completarse en máximo 1 día. Debe validarse compilación exitosa en emulador, simulador y dispositivo físico. Debe documentarse procedimiento para referencia futura.

Producciones esperadas :

- Android Studio instalado y configurado con SDK apropiados
- Emuladores de Android creados para múltiples versiones (API 21, 28, 33)
- Dispositivos físicos conectados y validados para depuración
- Proyecto compilado y ejecutado exitosamente en múltiples entornos
- Documentación de procedimiento de configuración

Competencias evaluadas : C8

Resolución de conflictos de dependencias en proyecto Android

Contexto : Proyecto Android existente presenta conflictos de compatibilidad entre versiones de librerías. Gradle no puede resolver dependencias debido a incompatibilidades entre versiones de bibliotecas de terceros. El desarrollador debe identificar conflictos, evaluar opciones de resolución y actualizar dependencias manteniendo estabilidad.

Restricciones : Debe resolverse sin romper funcionalidades existentes. Debe validarse compilación exitosa. Debe documentarse cambios realizados.

Producciones esperadas :

- Análisis de conflictos de dependencias identificados
- Evaluación de opciones de resolución (actualización, downgrade, alternativas)
- Archivo build.gradle actualizado con versiones compatibles
- Pruebas de compilación exitosas
- Documentación de cambios y justificación de versiones seleccionadas

Competencias evaluadas : C9

Comparación de enfoques nativo vs multiplataforma para aplicación móvil

Contexto : Empresa debe decidir si desarrollar aplicación móvil con enfoque nativo (Android/iOS separados) o multiplataforma (React Native/Flutter). Debe evaluarse impacto en rendimiento, mantenibilidad, costo y cronograma considerando requisitos específicos del proyecto.

Restricciones : Análisis debe completarse en máximo 3 semanas. Debe incluir pruebas de concepto de ambos enfoques. Debe validarse con equipo técnico.

Producciones esperadas :

- Matriz comparativa de enfoques nativo vs multiplataforma
- Análisis de rendimiento esperado para cada enfoque
- Evaluación de impacto en mantenibilidad y escalabilidad
- Estimación de costo y cronograma para cada enfoque
- Recomendación fundamentada con justificación técnica

Competencias evaluadas : C7

Evaluación y configuración de framework multiplataforma

Contexto : Equipo decide utilizar Flutter para desarrollo multiplataforma. Debe evaluarse framework considerando requisitos del proyecto, configurarse proyecto base, instalar dependencias y validar que entorno está listo para desarrollo.

Restricciones : Debe completarse en máximo 2 semanas. Debe incluir evaluación de al menos 2 frameworks alternativos. Debe validarse compilación en Android e iOS.

Producciones esperadas :

- Matriz de evaluación de frameworks multiplataforma
- Análisis de impacto en rendimiento y mantenibilidad
- Proyecto Flutter base configurado con dependencias necesarias
- Compilación exitosa en Android e iOS
- Documentación de decisión y configuración de proyecto

Competencias evaluadas : C7

Pruebas esperadas

■ Pruebas documentales

- Matriz de evaluación de plataformas con criterios ponderados y análisis comparativo
- Documento de análisis de cuota de mercado y distribución de usuarios por plataforma
- Especificación de restricciones técnicas de dispositivos identificadas
- Recomendación formal de selección de plataforma con justificación técnica
- Documentación de configuración de entorno de desarrollo
- Archivo build.gradle o Podfile con dependencias versionadas
- Procedimiento documentado de resolución de conflictos de dependencias
- Matriz comparativa de enfoques de desarrollo (nativo vs multiplataforma)
- Análisis de impacto en rendimiento, mantenibilidad y costo
- Documento de decisión técnica con alternativas consideradas

■ Pruebas de acción (en campo)

- Instalación y configuración de Android Studio con SDK apropiados
- Instalación y configuración de Xcode con SDK de iOS
- Creación de emuladores de Android para múltiples versiones de API
- Creación de simuladores de iOS para múltiples versiones

- Conexión de dispositivos físicos Android e iOS para depuración
- Compilación exitosa de proyecto base en múltiples entornos
- Ejecución de aplicación en emulador, simulador y dispositivo físico
- Resolución de conflictos de dependencias en Gradle o CocoaPods
- Actualización de dependencias manteniendo compatibilidad
- Validación de entorno de desarrollo con checklist de verificación

■ **Pruebas reflexivas**

- Análisis de criterios utilizados en evaluación de plataforma y justificación de ponderación
- Reflexión sobre trade-offs identificados entre enfoques nativo y multiplataforma
- Evaluación de decisión de plataforma considerando requisitos técnicos y de negocio
- Análisis de impacto de selección de framework en mantenibilidad futura
- Reflexión sobre desafíos encontrados en configuración de entorno y soluciones aplicadas
- Evaluación de completitud de configuración de entorno
- Análisis de lecciones aprendidas en resolución de conflictos de dependencias
- Reflexión sobre documentación de decisiones técnicas para referencia futura
- Evaluación de alineación de decisión de plataforma con objetivos de proyecto
- Análisis de riesgos técnicos identificados en selección de plataforma

Criterios de éxito (7 familias)

Familia	Definición	Ponderación
Conformité	Selección de plataforma cumple con requisitos técnicos y restricciones de negocio especificadas en proyecto	Matriz de evaluación demuestra que plataforma seleccionada cumple al menos 90% de criterios técnicos y de negocio
Qualité	Entorno de desarrollo está completamente configurado y validado para desarrollo productivo	Proyecto base compila exitosamente en emulador, simulador y dispositivo físico sin errores
Comunicación	Decisiones de plataforma y configuración están documentadas y comunicadas claramente a equipo	Documentación de decisión incluye justificación técnica, alternativas consideradas y validación con stakeholders
Pertinence	Evaluación de plataforma considera criterios relevantes para proyecto específico	Matriz de evaluación incluye al menos 6 criterios técnicos y de negocio específicos del proyecto
Cohérence	Selección de plataforma es coherente con requisitos funcionales y no funcionales del proyecto	Análisis demuestra alineación entre características de plataforma seleccionada y requisitos especificados
Traçabilité	Decisiones de configuración de entorno están documentadas y trazables a requisitos técnicos	Documentación de configuración incluye referencias a requisitos y justificación de cada decisión
Posture	Desarrollador demuestra rigor en evaluación de opciones y meticulosidad en configuración de entorno	Documentación refleja análisis exhaustivo de alternativas y validación sistemática de configuración

Umbral de validación : Promedio ≥ 10/20

Bloque 3 - Diseño de Arquitectura e Interfaz de Aplicaciones Móviles

Indicadores de desempeño por misión

Misión	Indicadores SMART
M10 — Diseñar la arquitectura de información de la aplicación	Diagrama de arquitectura de información documentado con todas las entidades y relaciones identificadas Modelo de datos normalizado sin redundancias innecesarias Flujos de navegación mapeados con transiciones claras entre pantallas Documentación de arquitectura completa con justificación de decisiones
M13 — Elegir un enfoque nativo para el desarrollo móvil	Patrón arquitectónico seleccionado aplicado consistentemente en toda la aplicación Separación clara de responsabilidades entre capas (Modelo, Vista, Controlador/Presentador) Código modular con bajo acoplamiento entre componentes Documentación de patrones implementados con ejemplos
M14 — Elegir un enfoque multiplataforma para el desarrollo móvil	Evaluación de frameworks multiplataforma documentada con análisis de impacto en rendimiento Configuración de proyecto base completada con estructura estándar Gestión de estado implementada con patrón seleccionado (Redux, BLoC, Provider) Optimización de flujos de estado validada con métricas de rendimiento

Situaciones de evaluación

Diseño de Arquitectura de Información para Aplicación de E-commerce

Contexto : Se requiere diseñar la arquitectura de información para una aplicación móvil de e-commerce que debe gestionar catálogo de productos, carrito de compras, historial de pedidos y perfiles de usuario. La aplicación debe funcionar en Android e iOS con sincronización de datos entre dispositivos.

Restricciones : Debe soportar múltiples idiomas, diferentes tamaños de pantalla, y mantener consistencia de datos en modo offline. El diseño debe ser escalable para agregar nuevas funcionalidades en futuras versiones.

Producciones esperadas :

- Diagrama de arquitectura de información con entidades principales (Producto, Carrito, Pedido, Usuario)
- Modelo de datos normalizado con relaciones y restricciones de integridad
- Mapa de flujos de navegación mostrando transiciones entre pantallas principales
- Documentación de decisiones arquitectónicas con justificación técnica

Competencias evaluadas : C6, C10

Implementación de Patrón MVVM en Aplicación de Redes Sociales

Contexto : Se debe implementar la lógica de negocio de una aplicación de redes sociales usando patrón MVVM. La aplicación debe gestionar feed de publicaciones, comentarios, likes y notificaciones en tiempo real. Se requiere separación clara entre lógica de presentación y lógica de negocio.

Restricciones : El código debe ser testeable, mantenible y seguir principios SOLID. Debe evitar acoplamiento entre capas y permitir reutilización de componentes lógicos.

Producciones esperadas :

- Implementación de ViewModels para cada pantalla principal
- Separación clara de responsabilidades entre Model, View y ViewModel
- Código modular con bajo acoplamiento entre componentes
- Documentación de patrones implementados con ejemplos de uso

Competencias evaluadas : C10, C15

Diseño de Interfaz Responsiva para Aplicación Bancaria

Contexto : Se debe diseñar la interfaz de usuario para una aplicación bancaria móvil que funcione en teléfonos y tablets con diferentes tamaños de pantalla. La interfaz debe seguir Material Design para Android y HIG para iOS, con énfasis en accesibilidad y seguridad visual.

Restricciones : Debe soportar múltiples tamaños de pantalla (4.5" a 7"), orientaciones (portrait y landscape), y cumplir con estándares de accesibilidad WCAG 2.1 AA. Los componentes deben ser reutilizables y documentados.

Producciones esperadas :

- Prototipos de alta fidelidad de pantallas principales en Figma o Sketch
- Especificación de componentes reutilizables con documentación de API
- Validación de accesibilidad con herramientas de análisis
- Pruebas de usabilidad con usuarios reales documentadas

Competencias evaluadas : C12, C13, C14, C16, C17, C18

Validación de Prototipos con Stakeholders para Aplicación de Fitness

Contexto : Se han creado prototipos de baja y alta fidelidad para una aplicación de fitness que incluye seguimiento de ejercicios, planes de entrenamiento y comunidad. Se requiere validar prototipos con stakeholders (clientes, usuarios potenciales) y recopilar feedback para iteración.

Restricciones : Debe obtener aprobación formal de stakeholders antes de iniciar desarrollo. Debe documentar todos los cambios solicitados y justificar decisiones de diseño. Debe validar que patrones de interacción son intuitivos.

Producciones esperadas :

- Presentación de wireframes y prototipos a stakeholders
- Documentación de feedback recibido y cambios solicitados
- Matriz de trazabilidad de requisitos a elementos de diseño
- Aprobación formal documentada de stakeholders

Competencias evaluadas : C16, C17, C18

Implementación de Navegación y Componentes en Aplicación de Viajes

Contexto : Se debe implementar la navegación y componentes visuales para una aplicación de viajes que incluye búsqueda de vuelos, hoteles, actividades y gestión de itinerarios. La aplicación debe tener flujos de navegación complejos con múltiples pantallas y paso de parámetros.

Restricciones : Debe implementar componentes reutilizables siguiendo Material Design y HIG. Debe gestionar correctamente el historial de navegación y permitir transiciones suaves. Debe ser accesible para usuarios con discapacidades.

Producciones esperadas :

- Implementación de componentes visuales reutilizables (tarjetas, listas, formularios)
- Sistema de navegación con flujos complejos implementado
- Animaciones de transición suaves entre pantallas
- Validación de accesibilidad con lectores de pantalla

Competencias evaluadas : C11, C12, C13, C14, C26

Pruebas esperadas

■ Pruebas documentales

- Diagrama de arquitectura de información con notación clara
- Modelo de datos normalizado documentado
- Especificación de componentes reutilizables con API documentada
- Prototipos de interfaz en herramientas de diseño (Figma, Sketch)
- Documentación de decisiones arquitectónicas y justificación técnica
- Matriz de trazabilidad de requisitos a elementos de diseño
- Especificación de patrones de navegación con flujos mapeados
- Documentación de patrones arquitectónicos implementados

■ Pruebas de acción (en campo)

- Implementación de componentes visuales siguiendo directrices de diseño
- Creación de layouts adaptativos para múltiples tamaños de pantalla
- Implementación de navegación entre pantallas con paso de parámetros
- Desarrollo de componentes reutilizables con documentación
- Implementación de patrones arquitectónicos (MVC, MVP, MVVM)
- Gestión de estado usando patrones seleccionados (Redux, BLoC, Provider)
- Validación de accesibilidad en interfaces implementadas
- Pruebas de usabilidad con usuarios reales

■ Pruebas reflexivas

- Análisis de decisiones arquitectónicas y alternativas consideradas
- Reflexión sobre escalabilidad y mantenibilidad del diseño
- Evaluación de feedback de usuarios y stakeholders
- Análisis de patrones implementados y su efectividad
- Reflexión sobre accesibilidad e inclusión en diseño
- Evaluación de consistencia de patrones en toda la aplicación
- Análisis de rendimiento de componentes y animaciones
- Reflexión sobre mejoras futuras basadas en experiencia

Criterios de éxito (7 familias)

Familia	Definición	Ponderación
Conformité	Arquitectura de información cumple con estándares de normalización de datos y no contiene redundancias innecesarias	Revisión de diagrama de datos contra tercera forma normal (3NF)
Qualité	Componentes visuales implementados siguen consistentemente directrices de Material Design o HIG según plataforma	Validación visual contra especificación de diseño en múltiples dispositivos
Comunicación	Documentación de arquitectura es clara, completa y comprensible para nuevos desarrolladores	Revisión de documentación por pares; capacidad de nuevos desarrolladores para entender y extender arquitectura
Pertinence	Patrón arquitectónico seleccionado es apropiado para requisitos de la aplicación y facilita separación de responsabilidades	Evaluación de coherencia entre patrón implementado y requisitos funcionales
Traçabilité	Cada componente de interfaz es rastreable a requisito funcional o de diseño documentado	Matriz de trazabilidad completa con referencias bidireccionales
Cohérence	Flujos de navegación son coherentes y predecibles, permitiendo usuarios entender cómo moverse entre pantallas	Pruebas de usabilidad con usuarios reales; tiempo para completar tareas de navegación
Posture	Desarrollador demuestra rigor en aplicación de patrones, atención al detalle en diseño y apertura a feedback de stakeholders	Consistencia de patrones en código; calidad de documentación; incorporación de feedback en iteraciones

Umbral de validación : Promedio ≥ 10/20

Bloque 4 - Desarrollo de Funcionalidades y Lógica de Negocio

Indicadores de desempeño por misión

Misión	Indicadores SMART
M05 — Analizar requisitos funcionales de la aplicación móvil	Porcentaje de requisitos funcionales documentados con criterios de aceptación claros Número de sesiones de validación con stakeholders completadas antes de desarrollo Cobertura de casos de uso documentados versus funcionalidades implementadas Trazabilidad de requisitos: cada requisito vinculado a al menos una historia de usuario
M09 — Traducir necesidades de negocio en funcionalidades móviles	Número de especificaciones técnicas generadas a partir de requisitos de negocio Claridad de especificaciones: porcentaje de desarrolladores que entienden especificación sin aclaraciones Tiempo de implementación versus estimación técnica Número de cambios de requisitos durante desarrollo (indicador de especificación incompleta)
M10 — Diseñar la arquitectura de información de la aplicación	Cobertura de arquitectura de información: porcentaje de datos mapeados en modelo Documentación de arquitectura: completitud de diagramas y especificaciones Adherencia a patrón arquitectónico: porcentaje de código que sigue patrón definido Escalabilidad: capacidad de agregar nuevas funcionalidades sin refactorización mayor

Situaciones de evaluación

Implementación de Persistencia Local con Sincronización

Contexto : Desarrollador debe implementar funcionalidad de almacenamiento local de datos de usuario en base de datos SQLite/Room (Android) o Core Data (iOS), incluyendo gestión de migraciones y sincronización con servidor cuando conectividad se restaura. Contexto realista: aplicación de lista de tareas que permite crear, editar y eliminar tareas offline, sincronizando cambios cuando hay conexión.

Restricciones : Debe manejar fallos de conectividad sin pérdida de datos, implementar validación de datos antes de guardar, documentar esquema de base de datos, gestionar conflictos de sincronización cuando usuario modifica datos offline.

Producciones esperadas :

- Código de modelo de datos con entidades y relaciones definidas
- Implementación de operaciones CRUD en base de datos local
- Mecanismo de sincronización con servidor
- Gestión de migraciones de esquema
- Pruebas unitarias de persistencia
- Documentación de esquema de datos

Competencias evaluadas : C19, C20, C21, C22

Integración de API REST con Manejo de Errores

Contexto : Desarrollador debe implementar cliente HTTP que consume API REST externa, procesando respuestas JSON, manejando errores de red y validando datos recibidos. Contexto realista: integración con API de clima que retorna datos meteorológicos, requiere manejo de timeouts, reintentos y fallback a datos en caché.

Restricciones : Debe implementar validación de datos recibidos, manejo de códigos de error HTTP, reintentos con backoff exponencial, abstracción de lógica de red en repositorio, logging de errores.

Producciones esperadas :

- Cliente HTTP configurado con interceptores
- Modelos de datos para respuestas JSON
- Implementación de repositorio con abstracción de red
- Manejo de errores con mensajes contextuales
- Pruebas de integración con API
- Documentación de endpoints consumidos

Competencias evaluadas : C20, C21, C22

Validación de Formulario Multipasos con Feedback Visual

Contexto : Desarrollador debe implementar formulario de registro de usuario con múltiples pasos, validación en tiempo real de campos, mensajes de error contextuales y feedback visual. Contexto realista: formulario de registro que valida email, contraseña, teléfono con requisitos específicos, mostrando errores mientras usuario escribe.

Restricciones : Debe implementar validaciones cliente y servidor, mensajes de error claros en español, gestión del teclado virtual, navegación accesible entre campos, prevención de envío duplicado.

Producciones esperadas :

- Componentes de formulario con validación integrada
- Validadores reutilizables para diferentes tipos de campo
- Mensajes de error contextuales y accesibles
- Gestión de estado del formulario
- Pruebas de validación
- Documentación de reglas de validación

Competencias evaluadas : C27, C28

Arquitectura de Lógica de Negocio con Patrón MVVM

Contexto : Desarrollador debe diseñar e implementar lógica de negocio compleja usando patrón MVVM, separando presentación de lógica, implementando gestión de estado y comunicación entre componentes. Contexto realista: funcionalidad de carrito de compras que calcula totales, aplica descuentos, valida inventario y sincroniza con servidor.

Restricciones : Debe aplicar patrón MVVM consistentemente, implementar gestión de estado observable, separar responsabilidades entre ViewModel y Vista, documentar flujos de datos, escribir pruebas unitarias de lógica.

Producciones esperadas :

- Implementación de ViewModel con lógica de negocio
- Modelos de datos estructurados
- Observables o Flows para gestión de estado
- Separación clara entre presentación y lógica
- Pruebas unitarias de ViewModel
- Documentación de flujos de datos

Competencias evaluadas : C10

Sincronización Offline-First con Resolución de Conflictos

Contexto : Desarrollador debe implementar estrategia de sincronización offline-first donde aplicación funciona completamente sin conexión, almacenando cambios localmente y sincronizando cuando hay conectividad, resolviendo conflictos cuando datos fueron modificados en servidor. Contexto realista: aplicación de notas que permite crear, editar y eliminar notas sin conexión, sincronizando cambios cuando se restaura conectividad.

Restricciones : Debe detectar cambios de conectividad, mantener cola de operaciones pendientes, resolver conflictos de edición concurrente, validar datos antes de sincronizar, notificar usuario de estado de sincronización.

Producciones esperadas :

- Implementación de detección de conectividad
- Cola de operaciones pendientes
- Lógica de resolución de conflictos
- Mecanismo de sincronización incremental
- Notificaciones de estado de sincronización
- Pruebas de sincronización en diferentes escenarios

Competencias evaluadas : C19, C20, C21, C22

Pruebas esperadas

■ Pruebas documentales

- Especificación técnica de funcionalidades implementadas
- Documentación de esquema de base de datos con diagramas entidad-relación
- Documentación de APIs consumidas incluyendo endpoints, parámetros y respuestas
- Especificación de reglas de validación con ejemplos
- Documentación de patrones arquitectónicos aplicados
- Matriz de trazabilidad de requisitos a código implementado
- Documentación de estrategia de sincronización offline-first
- Especificación de manejo de errores y recuperación

■ Pruebas de acción (en campo)

- Implementación de modelos de datos persistentes en base de datos local
- Codificación de operaciones CRUD en repositorio
- Implementación de cliente HTTP con manejo de errores
- Desarrollo de validadores de entrada de usuario
- Implementación de lógica de negocio usando patrón arquitectónico
- Codificación de mecanismo de sincronización con servidor
- Implementación de gestión de estado observable
- Desarrollo de pruebas unitarias de lógica de negocio
- Implementación de manejo de fallos de conectividad
- Codificación de resolución de conflictos de datos

■ Pruebas reflexivas

- Análisis de decisiones de diseño arquitectónico y justificación
- Reflexión sobre trade-offs entre rendimiento y mantenibilidad
- Evaluación de cobertura de pruebas y completitud de validaciones
- Análisis de vulnerabilidades de seguridad identificadas
- Reflexión sobre experiencia de usuario en manejo de errores

• Evaluación de escalabilidad de solución implementada
• Análisis de lecciones aprendidas en integración de APIs
• Reflexión sobre mejoras futuras en arquitectura

Criterios de éxito (7 familias)

Familia	Definición	Ponderación
Conformité	Implementación cumple con especificación técnica de requisitos funcionales	100% de criterios de aceptación validados en pruebas
Qualité	Código implementado sigue patrones arquitectónicos definidos y es mantenible	Cobertura de pruebas unitarias ≥ 80%, adherencia a patrón ≥ 90%
Comunicación	Documentación técnica es clara, completa y accesible para otros desarrolladores	Documentación cubre 100% de componentes principales, ejemplos de uso incluidos
Cohérence	Integración de datos y servicios es consistente con arquitectura definida	Todas las capas de aplicación siguen patrón definido, sin excepciones no documentadas
Traçabilité	Cada requisito funcional es trazable a código implementado y pruebas	Matriz de trazabilidad completa, 100% de requisitos vinculados a implementación
Pertinence	Validaciones de entrada protegen integridad de datos y experiencia de usuario	Todas las entradas validadas, mensajes de error contextuales, sin pérdida de datos
Posture	Desarrollador demuestra rigor en seguridad, validación y manejo de errores	Código sin vulnerabilidades identificadas, manejo exhaustivo de casos de error

Umbral de validación : Promedio ≥ 10/20

Bloque 5 - Integración de Servicios Avanzados y Funcionalidades Especializadas

Indicadores de desempeño por misión

Misión	Indicadores SMART
M01 — Verificar el comportamiento de la aplicación en Android	Porcentaje de casos de prueba funcionales ejecutados exitosamente en Android (≥95%) Número de defectos críticos identificados en pruebas funcionales (≤2 por release) Tiempo promedio de ejecución de suite de pruebas funcionales (<30 minutos) Cobertura de funcionalidades críticas en pruebas (≥90%)
M02 — Preparar compilaciones para la publicación de la aplicación	Número de compilaciones de producción generadas sin errores (100%) Tiempo de preparación de compilación de producción (<1 hora) Cumplimiento de requisitos de tienda de aplicaciones (100%) Número de intentos de publicación rechazados por tienda (0)
M03 — Aplicar principios de seguridad en el desarrollo de la aplicación	Número de vulnerabilidades OWASP Mobile identificadas y corregidas (100%) Porcentaje de código revisado por seguridad (≥80%) Número de incidentes de seguridad en producción (0) Tiempo promedio de corrección de vulnerabilidades (<48 horas)
M04 — Ejecutar pruebas de compatibilidad en distintos dispositivos	Número de dispositivos en matriz de pruebas (≥10 configuraciones) Porcentaje de defectos de compatibilidad identificados (≥95%) Tiempo promedio de ejecución de pruebas de compatibilidad (<2 horas) Número de versiones de SO cubiertas en pruebas (≥3 versiones por plataforma)

Situaciones de evaluación

Integración de Autenticación OAuth con Recuperación de Contraseña

Contexto : Se requiere implementar flujo de autenticación OAuth 2.0 en aplicación móvil existente, permitiendo login con Google y Apple, además de autenticación personalizada con recuperación de contraseña. La aplicación debe mantener sesiones seguras y renovar tokens automáticamente.

Restricciones : Debe cumplir con requisitos de seguridad OWASP Mobile, almacenar tokens de forma segura en Keystore/Keychain, implementar validación de certificados SSL y manejar expiración de sesiones.

Producciones esperadas :

- Implementación de flujo de login con OAuth 2.0
- Servicio de autenticación personalizada con hash seguro de contraseñas
- Mecanismo de recuperación de contraseña con validación de email
- Almacenamiento seguro de tokens en Keystore/Keychain
- Renovación automática de tokens antes de expiración
- Documentación de flujos de autenticación
- Pruebas unitarias de lógica de autenticación
- Análisis de seguridad del código de autenticación

Competencias evaluadas : C23, C24

Implementación de Sistema de Notificaciones Push Multiplataforma

Contexto : Se requiere implementar sistema de notificaciones push que funcione en Android e iOS, integrando Firebase Cloud Messaging y Apple Push Notification service. La aplicación debe manejar notificaciones en primer y segundo plano, gestionar permisos y proporcionar acciones personalizadas.

Restricciones : Debe solicitar permisos de notificación en tiempo de ejecución, manejar casos donde usuario deniega permisos, implementar gestión de canales de notificación en Android y validar que notificaciones se entregan correctamente.

Producciones esperadas :

- Integración de Firebase Cloud Messaging en Android
- Integración de Apple Push Notification service en iOS
- Implementación de solicitud de permisos de notificación
- Manejadores de notificaciones en primer plano
- Manejadores de notificaciones en segundo plano
- Acciones personalizadas en notificaciones
- Gestión de canales de notificación
- Pruebas de entrega de notificaciones
- Documentación de configuración de notificaciones

Competencias evaluadas : C25

Auditoría de Seguridad y Remediación de Vulnerabilidades OWASP Mobile

Contexto : Se requiere realizar auditoría de seguridad completa de aplicación móvil existente, identificar vulnerabilidades OWASP Mobile Top 10, implementar cifrado de datos sensibles y almacenamiento seguro. Documentar hallazgos y crear plan de remediación.

Restricciones : Debe identificar vulnerabilidades de inyección, almacenamiento inseguro, comunicación insegura, autenticación débil y criptografía insuficiente. Implementar cifrado TLS/SSL para comunicación, cifrado de datos en reposo y validación de certificados.

Producciones esperadas :

- Reporte de auditoría de seguridad detallado
- Identificación de vulnerabilidades OWASP Mobile
- Implementación de cifrado de datos en tránsito
- Implementación de cifrado de datos en reposo
- Almacenamiento seguro de credenciales en Keystore/Keychain
- Validación de certificados SSL
- Revisión de seguridad de código
- Plan de remediación con prioridades
- Pruebas de penetración básicas
- Documentación de controles de seguridad implementados

Competencias evaluadas : C34

Implementación de Animaciones e Interacciones Avanzadas

Contexto : Se requiere implementar conjunto de animaciones e interacciones avanzadas en aplicación móvil, incluyendo transiciones entre pantallas, micro-interacciones de feedback y animaciones complejas. Optimizar rendimiento para mantener 60 FPS sin impacto en batería.

Restricciones : Debe mantener fluidez de 60 FPS, minimizar consumo de batería, respetar preferencias de movimiento del usuario, implementar accesibilidad en animaciones y proporcionar feedback visual inmediato.

Producciones esperadas :

- Animaciones de transición entre pantallas
- Micro-interacciones de feedback visual
- Animaciones complejas y secuenciadas
- Optimización de rendimiento de animaciones
- Profiling de rendimiento
- Pruebas de fluidez (60 FPS)
- Análisis de consumo de batería
- Documentación de animaciones implementadas
- Pruebas de accesibilidad en animaciones

Competencias evaluadas : C26

Integración de Múltiples Servicios Avanzados en Aplicación Existente

Contexto : Se requiere integrar múltiples servicios avanzados en aplicación móvil existente: autenticación OAuth, notificaciones push, análisis de seguridad y animaciones. Asegurar que servicios funcionan correctamente juntos, manejar errores de integración y documentar arquitectura.

Restricciones : Debe manejar fallos de servicios individuales sin afectar resto de aplicación, implementar reintentos inteligentes, proporcionar fallback a funcionalidad alternativa y comunicar errores claramente a usuarios.

Producciones esperadas :

- Arquitectura de integración de servicios
- Implementación de patrones de comunicación asincrónica
- Manejo de errores en integración de servicios
- Reintentos inteligentes con backoff exponencial
- Fallback a funcionalidad alternativa
- Gestión de dependencias entre servicios
- Pruebas de integración de servicios
- Documentación de arquitectura de servicios
- Análisis de rendimiento de integración

Competencias evaluadas : C23, C24, C25, C26, C34

Pruebas esperadas

■ Pruebas documentales

- Documentación de flujos de autenticación con diagramas de secuencia
- Especificación técnica de integración de servicios avanzados
- Reporte de auditoría de seguridad con vulnerabilidades identificadas
- Documentación de configuración de notificaciones push
- Especificación de animaciones e interacciones implementadas
- Matriz de trazabilidad de requisitos de seguridad
- Documentación de decisiones de arquitectura
- Procedimientos de almacenamiento seguro de credenciales
- Guía de implementación de cifrado
- Documentación de gestión de permisos

■ Pruebas de acción (en campo)

- Implementación de flujo de autenticación OAuth 2.0 funcional
- Integración de Firebase Cloud Messaging en Android
- Integración de Apple Push Notification service en iOS
- Implementación de cifrado de datos en tránsito y en reposo
- Almacenamiento seguro de tokens en Keystore/Keychain
- Implementación de animaciones de transición entre pantallas
- Ejecución de auditoría de seguridad completa
- Corrección de vulnerabilidades OWASP Mobile identificadas
- Implementación de validación de certificados SSL
- Configuración de gestión de permisos en tiempo de ejecución

■ Pruebas reflexivas

- Análisis de decisiones de seguridad tomadas en implementación
- Reflexión sobre trade-offs entre seguridad y rendimiento

• Evaluación de impacto de animaciones en experiencia de usuario
• Análisis de vulnerabilidades identificadas y lecciones aprendidas
• Reflexión sobre mejoras en arquitectura de servicios
• Evaluación de efectividad de flujos de autenticación
• Análisis de feedback de usuarios sobre interacciones
• Reflexión sobre mejoras en gestión de errores
• Evaluación de cumplimiento de requisitos de seguridad
• Análisis de impacto de cambios en experiencia de usuario

Criterios de éxito (7 familias)

Familia	Definición	Ponderación
Conformité	Cumplimiento de requisitos de seguridad OWASP Mobile Top 10 en todas las integraciones de servicios	100% de vulnerabilidades identificadas corregidas antes de publicación
Qualité	Calidad técnica de implementación de servicios avanzados con código limpio y bien documentado	Cobertura de pruebas ≥80%, complejidad ciclomática <10, documentación completa de APIs
Comunicación	Comunicación clara de errores y estados a usuarios en integraciones de servicios	Mensajes de error comprensibles en 100% de casos de fallo, feedback visual en todas las interacciones
Pertinence	Pertinencia de servicios integrados respecto a requisitos de negocio y necesidades de usuarios	Validación de requisitos con stakeholders, feedback positivo de usuarios en pruebas
Cohérence	Coherencia de arquitectura de servicios con patrones de diseño establecidos	Consistencia en patrones de integración, reutilización de componentes, documentación de decisiones
Traçabilité	Trazabilidad de requisitos de seguridad desde especificación hasta implementación y validación	Matriz de trazabilidad completa, auditoría de seguridad documentada, pruebas de seguridad ejecutadas
Posture	Postura responsable en manejo de datos sensibles y seguridad de aplicación	Nunca hardcodear credenciales, almacenamiento seguro de tokens, auditoría de acceso a datos sensibles

Umbral de validación : Promedio ≥ 10/20

Bloque 6 - Pruebas, Depuración y Optimización de Aplicaciones Móviles

Indicadores de desempeño por misión

Misión	Indicadores SMART
M01 — Verificar el comportamiento de la aplicación en Android	Porcentaje de casos de prueba funcionales ejecutados exitosamente en Android Número de defectos identificados durante pruebas funcionales Cobertura de código alcanzada en pruebas unitarias Tiempo promedio de ejecución de suite de pruebas
M04 — Ejecutar pruebas de compatibilidad en distintos dispositivos	Número de dispositivos en matriz de pruebas cubiertos Porcentaje de funcionalidades validadas en matriz de dispositivos Número de problemas de compatibilidad identificados Tiempo de resolución promedio de defectos de compatibilidad
M03 — Aplicar principios de seguridad en el desarrollo de la aplicación	Número de vulnerabilidades OWASP identificadas Porcentaje de código revisado por seguridad Número de defectos de seguridad corregidos Cumplimiento de estándares de seguridad móvil

Situaciones de evaluación

Escritura de pruebas unitarias para nueva funcionalidad

Contexto : Se ha implementado un nuevo módulo de lógica de negocio en la aplicación móvil. El desarrollador debe escribir pruebas unitarias exhaustivas que validen el comportamiento correcto del módulo en diferentes escenarios, incluyendo casos normales, casos límite y manejo de errores.

Restricciones : Las pruebas deben alcanzar mínimo 80% de cobertura de código. Deben utilizar framework de testing apropiado (JUnit para Android, XCTest para iOS). Deben ejecutarse en menos de 5 segundos.

Producciones esperadas :

- Suite de pruebas unitarias con mínimo 10 casos de prueba
- Reporte de cobertura de código
- Documentación de casos de prueba y propósito
- Evidencia de ejecución exitosa de pruebas

Competencias evaluadas : C31

Depuración de error complejo en aplicación móvil

Contexto : Se ha reportado un error crítico en producción donde la aplicación se bloquea bajo condiciones específicas de red. El desarrollador debe reproducir el error, analizar logs y trazas, identificar causa raíz y validar que la solución resuelve el problema sin crear nuevos errores.

Restricciones : Debe reproducir error de forma consistente. Debe documentar pasos de reproducción. Debe validar solución en múltiples escenarios. Debe verificar que no hay regresiones.

Producciones esperadas :

- Documentación de pasos de reproducción
- Análisis de logs y trazas de error
- Identificación de causa raíz
- Código de solución con explicación
- Evidencia de validación de solución

Competencias evaluadas : C29, C30

Ejecución de pruebas de compatibilidad en matriz de dispositivos

Contexto : Antes de publicar nueva versión de aplicación, se debe validar compatibilidad en matriz de dispositivos incluyendo diferentes versiones de Android/iOS, tamaños de pantalla y configuraciones de hardware. El desarrollador debe ejecutar pruebas, documentar problemas encontrados y priorizar defectos.

Restricciones : Matriz mínima de 5 dispositivos diferentes. Debe incluir versiones antiguas y recientes de SO. Debe documentar cada problema con pasos de reproducción. Debe priorizar defectos según severidad.

Producciones esperadas :

- Plan de pruebas de compatibilidad
- Reporte de ejecución de pruebas
- Documentación de problemas de compatibilidad
- Matriz de priorización de defectos
- Evidencia de pruebas en múltiples dispositivos

Competencias evaluadas : C33

Implementación de pruebas automatizadas de interfaz de usuario

Contexto : Se debe automatizar pruebas de flujo crítico de usuario (login, navegación principal, operación clave). El desarrollador debe escribir pruebas que simulen interacciones de usuario, validen elementos visuales y verifiquen comportamiento correcto en diferentes escenarios.

Restricciones : Debe utilizar framework apropiado (Espresso para Android, XCUITest para iOS). Debe cubrir flujo completo de usuario. Debe ejecutarse en menos de 30 segundos. Debe ser mantenible y reutilizable.

Producciones esperadas :

- Suite de pruebas de interfaz de usuario
- Documentación de escenarios de prueba
- Evidencia de ejecución exitosa
- Reporte de cobertura de flujos de usuario
- Código limpio y bien documentado

Competencias evaluadas : C32

Análisis de seguridad y revisión de vulnerabilidades OWASP

Contexto : Se debe realizar análisis de seguridad de aplicación móvil identificando vulnerabilidades OWASP Mobile, revisando código para patrones inseguros y validando implementación de medidas de seguridad. El desarrollador debe documentar hallazgos y proponer soluciones.

Restricciones : Debe cubrir OWASP Mobile Top 10. Debe incluir análisis estático y revisión manual. Debe documentar cada vulnerabilidad con severidad. Debe proponer soluciones concretas.

Producciones esperadas :

- Reporte de análisis de seguridad
- Identificación de vulnerabilidades OWASP
- Análisis de código inseguro
- Propuestas de solución
- Plan de remediación

Competencias evaluadas : C34

Pruebas esperadas

■ Pruebas documentales

- Reporte de cobertura de pruebas unitarias con métricas de cobertura
- Documentación de casos de prueba con descripción de escenarios
- Logs de ejecución de pruebas automatizadas
- Reporte de compatibilidad en matriz de dispositivos
- Documentación de defectos con pasos de reproducción
- Análisis de seguridad con identificación de vulnerabilidades
- Plan de pruebas de compatibilidad
- Matriz de priorización de defectos
- Reporte de análisis de logs y trazas de error

■ Pruebas de acción (en campo)

- Escribir y ejecutar pruebas unitarias en framework de testing
- Utilizar debugger para inspeccionar estado de aplicación
- Ejecutar pruebas en emuladores y dispositivos físicos
- Reproducir errores de forma sistemática
- Analizar logs y volcados de memoria
- Implementar soluciones de depuración
- Ejecutar pruebas de compatibilidad en matriz de dispositivos
- Documentar problemas de compatibilidad
- Realizar análisis estático de seguridad
- Revisar código para vulnerabilidades

■ Pruebas reflexivas

- Reflexión sobre efectividad de estrategia de testing utilizada
- Análisis de patrones de errores recurrentes
- Evaluación de cobertura de pruebas y áreas de mejora
- Reflexión sobre técnicas de depuración más efectivas
- Análisis de causas raíz de defectos de compatibilidad
- Evaluación de vulnerabilidades de seguridad identificadas
- Reflexión sobre mejoras en proceso de testing
- Análisis de lecciones aprendidas de errores

• Evaluación de efectividad de medidas de seguridad

Criterios de éxito (7 familias)

Familia	Definición	Ponderación
Conformité	Las pruebas unitarias cumplen con estándares de cobertura mínima del 80% de código	Reporte de cobertura de código generado por herramienta de análisis
Qualité	Las pruebas automatizadas validan correctamente comportamiento de aplicación en múltiples escenarios	Porcentaje de casos de prueba que pasan exitosamente sin falsos positivos
Comunicación	La documentación de defectos es clara, completa y contiene pasos de reproducción precisos	Evaluación de claridad y completitud de reportes de defectos por equipo de QA
Pertinence	Los casos de prueba son relevantes y validan funcionalidades críticas de la aplicación	Alineación de casos de prueba con requisitos funcionales documentados
Traçabilité	Cada defecto identificado está trazable a requisito funcional y validado en pruebas	Matriz de trazabilidad entre defectos, requisitos y casos de prueba
Cohérence	Las pruebas son consistentes en metodología, nomenclatura y estructura	Revisión de código de pruebas para verificar consistencia de patrones
Posture	El desarrollador demuestra rigor, atención al detalle y mentalidad de calidad en testing	Evaluación de actitud hacia testing, búsqueda proactiva de casos de prueba y documentación completa

Umbral de validación : Promedio ≥ 10/20

Bloque 7 - Publicación, Actualización y Mantenimiento de Aplicaciones Móviles

Indicadores de desempeño por misión

Misión	Indicadores SMART
M02 — Preparar compilaciones para la publicación de la aplicación	Compilación de producción se genera sin errores y con todas las optimizaciones habilitadas Artefactos distribuidores (APK, AAB, IPA) se generan correctamente y pasan validación de integridad Claves de firma se almacenan de forma segura y acceso se audita correctamente Aplicación publicada funciona correctamente en dispositivos reales sin errores críticos

Situaciones de evaluación

Preparación de compilación de producción para Android

Contexto : Desarrollador debe preparar compilación de producción de aplicación Android para publicación en Google Play Store. Aplicación ha completado todas las pruebas y está lista para distribución. Desarrollador debe configurar parámetros de compilación, generar APK/AAB optimizado y validar artefacto.

Restricciones : Debe habilitarse todas las optimizaciones de compilador, código de depuración debe estar deshabilitado, artefacto debe pasar validación de Google Play Store, claves de firma deben manejarse de forma segura

Producciones esperadas :

- Archivo de configuración de compilación (build.gradle) con parámetros de producción
- APK o AAB generado y validado
- Documentación de parámetros de compilación utilizados
- Checklist de validación completado

Competencias evaluadas : C35

Gestión segura de claves de firma para publicación

Contexto : Equipo de desarrollo necesita establecer procedimiento seguro para gestión de claves de firma utilizadas para publicar aplicaciones en tiendas. Desarrollador debe documentar procedimiento de almacenamiento, acceso y auditoría de claves.

Restricciones : Claves nunca deben estar en repositorio de código, acceso debe estar limitado a personas autorizadas, debe existir auditoría de quién accede a claves, procedimiento debe cumplir políticas de seguridad de organización

Producciones esperadas :

- Documentación de procedimiento de gestión de claves
- Configuración de almacenamiento seguro de claves
- Registro de auditoría de acceso a claves
- Validación de que claves están protegidas

Competencias evaluadas : C35

Validación de cumplimiento de requisitos de tienda antes de publicación

Contexto : Antes de publicar aplicación en Google Play Store o App Store, desarrollador debe validar que aplicación cumple todos los requisitos de tienda. Debe revisar políticas de privacidad, permisos, accesibilidad y contenido.

Restricciones : Aplicación debe cumplir todas las directrices de tienda, debe documentarse cualquier cambio realizado para cumplimiento, debe obtenerse aprobación antes de publicación

Producciones esperadas :

- Checklist de cumplimiento de requisitos de tienda
- Documentación de cambios realizados para cumplimiento
- Evidencia de validación de accesibilidad
- Aprobación formal de cumplimiento

Competencias evaluadas : C35

Publicación de aplicación en tienda de aplicaciones

Contexto : Desarrollador debe publicar aplicación en Google Play Store o App Store. Debe cargar artefactos, configurar información de aplicación, escribir descripción y cambios, y monitorear proceso de publicación.

Restricciones : Artefactos deben estar correctamente firmados, información de aplicación debe ser completa y precisa, descripción debe ser clara y atractiva, debe cumplirse todos los requisitos de tienda

Producciones esperadas :

- Aplicación publicada en tienda de aplicaciones
- Descripción de aplicación y notas de versión
- Capturas de pantalla y previsualizaciones
- Confirmación de publicación exitosa

Competencias evaluadas : C35

Monitoreo y mantenimiento post-publicación

Contexto : Después de publicar aplicación, desarrollador debe monitorear comportamiento en producción, recopilar feedback de usuarios y planificar actualizaciones. Debe identificar problemas, analizar métricas y planificar mejoras.

Restricciones : Debe monitorearse crashes y errores, debe analizarse feedback de usuarios, debe identificarse problemas de compatibilidad, debe planificarse actualizaciones basadas en datos

Producciones esperadas :

- Reporte de métricas de uso y crashes
- Análisis de feedback de usuarios
- Plan de actualizaciones y mejoras
- Documentación de problemas identificados

Competencias evaluadas : C35

Pruebas esperadas

■ Pruebas documentales

- Archivo de configuración de compilación (build.gradle, Xcode project settings) con parámetros de producción
- Documentación de procedimiento de gestión de claves de firma
- Checklist de cumplimiento de requisitos de tienda

- Notas de versión y descripción de aplicación publicada
- Reporte de métricas post-publicación
- Registro de auditoría de acceso a claves de firma
- Documentación de decisiones de versionado

■ Pruebas de acción (en campo)

- Configuración de parámetros de compilación de producción
- Generación de APK/AAB/IPA optimizado
- Firma de artefactos con claves de firma
- Carga de artefactos en consola de desarrollador
- Publicación de aplicación en tienda
- Monitoreo de proceso de publicación
- Validación de aplicación publicada en dispositivos reales
- Implementación de procedimiento de gestión de claves

■ Pruebas reflexivas

- Reflexión sobre decisiones de configuración de compilación y justificación
- Análisis de riesgos de seguridad en manejo de claves
- Evaluación de cumplimiento de requisitos de tienda
- Análisis de feedback de usuarios post-publicación
- Reflexión sobre lecciones aprendidas en proceso de publicación
- Evaluación de efectividad de procedimientos de seguridad
- Análisis de impacto de actualizaciones en usuarios

Criterios de éxito (7 familias)

Familia	Definición	Ponderación
Conformité	Compilación de producción cumple todos los requisitos técnicos de tienda (Google Play Store, App Store) incluyendo firma correcta, optimizaciones habilitadas y ausencia de código de depuración	Artefacto pasa validación de tienda sin errores, compilación se genera sin warnings críticos
Qualité	Artefactos distribuidores se generan correctamente, son validados antes de publicación y funcionan sin errores en dispositivos reales	Aplicación publicada funciona correctamente, no hay crashes críticos en primeras 24 horas post-publicación
Comunicación	Descripción de aplicación, notas de versión y cambios se comunican de forma clara y atractiva a usuarios	Descripción es comprensible, cambios están documentados claramente, usuarios entienden qué cambió
Pertinence	Decisiones de versionado y actualización son relevantes para necesidades de usuarios y negocio	Versiones se publican según plan, actualizaciones abordan problemas identificados, feedback de usuarios se incorpora
Traçabilité	Acceso a claves de firma se audita, procedimientos de publicación se documentan, cambios se rastrean	Registro de auditoría completo, documentación de procedimientos disponible, historial de publicaciones disponible

Familia	Definición	Ponderación
Posture	Responsabilidad en manejo de credenciales, vigilancia de seguridad, cumplimiento de requisitos de tienda	Claves se almacenan de forma segura, no hay brechas de seguridad, cumplimiento de tienda es verificado antes de publicación
Cohérence	Procedimientos de publicación son consistentes, versionado sigue estándares, actualizaciones mantienen compatibilidad	Procedimientos se aplican consistentemente, versionado sigue semántica, compatibilidad hacia atrás se mantiene

Umbral de validación : Promedio ≥ 10/20

Bloque 8 - Seguridad, Internacionalización y Gestión del Ciclo de Vida

Indicadores de desempeño por misión

Misión	Indicadores SMART
M03 — Aplicar principios de seguridad en el desarrollo de la aplicación	Número de vulnerabilidades OWASP Mobile identificadas y documentadas en análisis de seguridad Porcentaje de datos sensibles protegidos con cifrado implementado correctamente Cobertura de revisión de seguridad de código (líneas de código revisadas vs. total) Tiempo medio de remediación de vulnerabilidades críticas identificadas
M02 — Preparar compilaciones para la publicación de la aplicación	Número de compilaciones de producción generadas sin errores de firma Tiempo de preparación de artefactos de distribución (APK, IPA, AAB) Cumplimiento de requisitos de tiendas de aplicaciones en primer intento Número de incidentes de seguridad relacionados con gestión de claves
M04 — Ejecutar pruebas de compatibilidad en distintos dispositivos	Cobertura de matriz de dispositivos probados (número de dispositivos vs. total planificado) Número de problemas de compatibilidad identificados y documentados Tiempo medio de resolución de defectos de compatibilidad Porcentaje de defectos críticos identificados antes de publicación

Situaciones de evaluación

Auditoría de Seguridad de Aplicación Móvil en Producción

Contexto : Un equipo de desarrollo debe realizar una auditoría de seguridad completa de una aplicación móvil que ya está en producción. La aplicación maneja datos sensibles de usuarios (credenciales, información financiera) y se ha reportado un incidente de seguridad menor. Se requiere identificar vulnerabilidades OWASP Mobile, evaluar implementación de cifrado, revisar almacenamiento de datos y documentar hallazgos con recomendaciones de remediación.

Restricciones : Auditoría debe completarse sin interrumpir servicio en producción. Debe cumplir con estándares de seguridad de la industria. Debe documentarse de forma que sea comprensible para stakeholders no técnicos.

Producciones esperadas :

- Reporte de auditoría de seguridad con vulnerabilidades identificadas
- Matriz de riesgos con severidad y impacto de cada vulnerabilidad
- Plan de remediación con prioridades y cronograma
- Documentación de controles de seguridad implementados
- Recomendaciones para mejora continua de seguridad

Competencias evaluadas : Identificación de vulnerabilidades OWASP Mobile, Implementación de cifrado en aplicaciones móviles, Almacenamiento seguro de datos, Revisión de seguridad de código

Implementación de Soporte Multiidioma para Mercados Globales

Contexto : Una aplicación móvil exitosa en mercado español debe expandirse a 5 nuevos mercados (inglés, francés, alemán, portugués, japonés). Se requiere implementar sistema completo de internacionalización incluyendo traducción de interfaz, adaptación de formatos regionales (fechas, moneda, números) y pruebas de localización con usuarios reales de cada región.

Restricciones : Tamaño de aplicación no debe aumentar significativamente. Debe mantener rendimiento en dispositivos con recursos limitados. Debe soportar cambio dinámico de idioma sin reinicio de aplicación.

Producciones esperadas :

- Arquitectura de internacionalización documentada
- Archivos de recursos de idioma para todos los idiomas soportados
- Implementación de adaptación regional (fechas, moneda, formatos)
- Pruebas de localización con usuarios de cada región
- Documentación de guía de localización para traductores

Competencias evaluadas : Soporte multiidioma en aplicaciones móviles, Adaptación regional y cultural, Diseño sensible a cultura, Gestión de contenido multiidioma

Planificación de Actualización Mayor de Aplicación con Cambios de Dependencias

Contexto : Equipo debe planificar actualización mayor de aplicación que incluye: actualización de SDK de Android/iOS a versiones recientes, migración de librerías deprecadas, cambios en API de autenticación, y mejoras de seguridad. Actualización afectará a millones de usuarios activos. Se requiere planificar estrategia de lanzamiento, comunicación a usuarios, soporte de versiones antiguas y plan de rollback.

Restricciones : Debe minimizar impacto en usuarios. Debe mantener compatibilidad hacia atrás donde sea posible. Debe comunicarse claramente cambios y beneficios. Debe tener plan de contingencia para rollback rápido.

Producciones esperadas :

- Plan de actualización con cronograma detallado
- Documentación de cambios y mejoras para usuarios
- Estrategia de comunicación a usuarios
- Plan de soporte para versiones antiguas
- Plan de rollback y criterios de activación
- Documentación técnica de cambios de dependencias

Competencias evaluadas : Planificación de actualizaciones, Mantenimiento y soporte de aplicación, Evolución de dependencias, Gestión del ciclo de vida

Preparación de Compilación de Producción con Gestión Segura de Claves

Contexto : Equipo debe preparar compilación de producción de aplicación para publicación en Google Play y App Store. Requiere configurar parámetros de optimización, generar artefactos de distribución (APK, AAB para Android; IPA para iOS), gestionar claves de firma de forma segura, y validar cumplimiento de requisitos de tiendas. Claves de firma son activos críticos que deben protegerse.

Restricciones : Claves de firma deben almacenarse de forma segura sin exponerse en repositorio. Acceso a claves debe ser limitado y auditado. Compilación debe ser reproducible. Debe cumplir con requisitos específicos de Google Play y App Store.

Producciones esperadas :

- Compilación de producción optimizada
- Artefactos de distribución (APK, AAB, IPA) generados correctamente
- Documentación de procedimiento de firma segura
- Checklist de cumplimiento de requisitos de tiendas
- Registro de auditoría de acceso a claves de firma
- Documentación de procedimiento de rollback

Competencias evaluadas : Configuración de compilaciones de producción, Generación de artefactos de distribución, Gestión de claves de firma

Ejecución de Pruebas de Compatibilidad en Matriz de Dispositivos Diversa

Contexto : Antes de publicación de nueva versión, equipo debe ejecutar pruebas de compatibilidad exhaustivas en matriz de 20+ dispositivos con diferentes características (tamaños de pantalla, versiones de SO, capacidades de hardware). Se requiere documentar problemas encontrados, priorizar defectos según severidad e impacto, y validar que aplicación funciona correctamente en todos los dispositivos.

Restricciones : Pruebas deben completarse en tiempo limitado. Algunos dispositivos pueden no estar disponibles físicamente. Debe identificarse problemas críticos antes de publicación. Debe documentarse claramente para equipo de desarrollo.

Producciones esperadas :

- Plan de pruebas de compatibilidad con matriz de dispositivos
- Reporte de ejecución de pruebas con resultados por dispositivo
- Documentación de problemas de compatibilidad identificados
- Matriz de priorización de defectos
- Plan de remediación de defectos críticos
- Validación de correcciones en dispositivos afectados

Competencias evaluadas : Planificación de pruebas de compatibilidad, Ejecución de pruebas en matriz de dispositivos, Documentación de problemas de compatibilidad, Priorización de defectos

Pruebas esperadas

■ Pruebas documentales

- Reporte de auditoría de seguridad con vulnerabilidades identificadas y plan de remediación
- Documentación de arquitectura de internacionalización y guía de localización
- Plan de actualización con cronograma, comunicación a usuarios y plan de rollback
- Documentación de procedimiento de firma segura y gestión de claves
- Reporte de pruebas de compatibilidad con matriz de dispositivos y problemas identificados
- Matriz de trazabilidad de requisitos de seguridad
- Especificación de criterios de aceptación para seguridad e internacionalización
- Documentación de decisiones técnicas de seguridad

■ **Pruebas de acción (en campo)**

- Realizar auditoría de seguridad identificando vulnerabilidades OWASP Mobile
- Implementar cifrado de datos sensibles en almacenamiento local
- Configurar almacenamiento seguro de tokens y credenciales
- Implementar soporte multiidioma en interfaz de usuario
- Adaptar formatos regionales (fechas, moneda, números)
- Ejecutar pruebas de localización con usuarios de diferentes regiones
- Preparar compilación de producción con parámetros optimizados
- Generar artefactos de distribución (APK, AAB, IPA)
- Gestionar claves de firma de forma segura
- Ejecutar pruebas de compatibilidad en matriz de dispositivos
- Documentar problemas de compatibilidad y priorizar defectos
- Planificar actualización mayor con comunicación a usuarios

■ **Pruebas reflexivas**

- Reflexión sobre decisiones de seguridad tomadas y alternativas consideradas
- Análisis de impacto de vulnerabilidades identificadas en usuarios
- Evaluación de efectividad de medidas de seguridad implementadas
- Reflexión sobre experiencia de usuarios en diferentes mercados y culturas
- Análisis de feedback de usuarios de diferentes regiones
- Evaluación de estrategia de internacionalización y mejoras futuras
- Reflexión sobre lecciones aprendidas en gestión de ciclo de vida
- Análisis de impacto de cambios de dependencias en aplicación
- Evaluación de procedimientos de seguridad en gestión de claves
- Reflexión sobre problemas de compatibilidad encontrados y prevención futura

Criterios de éxito (7 familias)

Familia	Definición	Ponderación
Conformité	Aplicación cumple con estándares de seguridad OWASP Mobile Top 10 y requisitos de protección de datos	Auditoría de seguridad identifica 0 vulnerabilidades críticas, máximo 2 vulnerabilidades altas
Qualité	Implementación de cifrado y almacenamiento seguro de datos es técnicamente correcta y sigue mejores prácticas	Revisión de código de seguridad aprueba implementación sin hallazgos críticos
Comunicación	Documentación de seguridad, internacionalización y ciclo de vida es clara y comprensible para stakeholders técnicos y no técnicos	Documentación es aprobada por stakeholders, usuarios pueden entender cambios y beneficios
Pertinence	Medidas de seguridad implementadas son relevantes a riesgos reales identificados en aplicación	Cada medida de seguridad está vinculada a vulnerabilidad específica o requisito de negocio
Cohérence	Estrategia de internacionalización es coherente en toda la aplicación con adaptación consistente de idiomas y formatos regionales	Pruebas de localización validan consistencia de traducción y formatos en todos los idiomas soportados
Traçabilité	Todas las decisiones de seguridad, internacionalización y ciclo de vida están documentadas con justificación clara	Matriz de trazabilidad vincula requisitos a implementación a pruebas

Familia	Definición	Ponderación
Posture	Equipo demuestra vigilancia constante de seguridad, consideración de usuarios globales y responsabilidad a largo plazo en mantenimiento	Equipo participa en actualizaciones de seguridad, realiza pruebas con usuarios reales de diferentes regiones, planifica mantenimiento a largo plazo

Umbral de validación : Promedio ≥ 10/20

Bloque 9 - Documentación, Control de Versiones e Integración Continua

Indicadores de desempeño por misión

Misión	Indicadores SMART
M01 — Verificar el comportamiento de la aplicación en Android	Porcentaje de pruebas funcionales documentadas en repositorio de control de versiones Tiempo de ejecución de suite de pruebas en pipeline CI/CD Número de defectos identificados antes de publicación mediante pruebas automatizadas Cobertura de código de pruebas funcionales en Android
M02 — Preparar compilaciones para la publicación de la aplicación	Número de compilaciones de producción generadas exitosamente en pipeline CI/CD Tiempo desde commit hasta generación de artefacto distribuidora Porcentaje de compilaciones que pasan validación de firma Documentación de procedimiento de compilación completada y actualizada
M03 — Aplicar principios de seguridad en el desarrollo de la aplicación	Número de vulnerabilidades OWASP Mobile identificadas mediante análisis estático Porcentaje de código revisado para seguridad antes de publicación Tiempo de resolución de vulnerabilidades de seguridad identificadas Documentación de revisión de seguridad completada
M04 — Ejecutar pruebas de compatibilidad en distintos dispositivos	Número de dispositivos en matriz de pruebas de compatibilidad Porcentaje de pruebas ejecutadas automáticamente en pipeline CI/CD Número de problemas de compatibilidad documentados Tiempo de ejecución de suite de pruebas de compatibilidad

Situaciones de evaluación

Configuración de Pipeline CI/CD para Proyecto Móvil

Contexto : Se inicia un nuevo proyecto de desarrollo de aplicación móvil Android. El equipo necesita establecer un pipeline de integración continua que ejecute pruebas automáticamente en cada commit, genere reportes de cobertura y publique artefactos en repositorio de distribución.

Restricciones : El pipeline debe ejecutarse en menos de 15 minutos, debe soportar múltiples ramas de desarrollo, debe mantener secretos seguros (claves de firma, credenciales), debe generar notificaciones de fallos.

Producciones esperadas :

- Archivo de configuración de pipeline (GitHub Actions, GitLab CI, Jenkins)
- Documentación de pasos del pipeline
- Configuración de secretos y variables de entorno
- Reportes de ejecución del pipeline
- Documentación de cómo agregar nuevas etapas al pipeline

Documentación de Arquitectura e Integración en Repositorio

Contexto : Un proyecto de aplicación móvil tiene arquitectura compleja con múltiples capas, patrones de diseño y decisiones técnicas importantes. Se requiere documentar esta arquitectura de forma clara para que nuevos desarrolladores puedan entender la estructura y contribuir efectivamente.

Restricciones : La documentación debe ser clara y accesible, debe incluir diagramas visuales, debe explicar decisiones arquitectónicas, debe mantenerse actualizada con cambios de código.

Producciones esperadas :

- Documentación de arquitectura en formato Markdown
- Diagramas de arquitectura (C4, UML)
- Documentación de patrones de diseño utilizados
- Guía de contribución para nuevos desarrolladores
- Documentación de API interna
- Archivo ADR (Architecture Decision Record) para decisiones importantes

Implementación de Análisis de Calidad de Código en Pipeline

Contexto : El equipo de desarrollo desea mejorar la calidad del código implementando análisis estático automático. Se necesita configurar herramientas de análisis de código que se ejecuten en cada commit, generen reportes de calidad y bloqueen merges si la calidad no cumple estándares.

Restricciones : El análisis debe ejecutarse rápidamente (menos de 5 minutos), debe detectar vulnerabilidades de seguridad, debe medir cobertura de pruebas, debe ser configurable para diferentes proyectos.

Producciones esperadas :

- Configuración de herramienta de análisis estático (SonarQube, Lint, Detekt)
- Reglas de calidad personalizadas para proyecto
- Integración en pipeline CI/CD
- Dashboard de métricas de calidad
- Documentación de cómo interpretar reportes de calidad
- Procedimiento de resolución de problemas de calidad

Gestión de Versiones y Etiquetado en Repositorio

Contexto : Un proyecto móvil necesita gestionar múltiples versiones de la aplicación en producción. Se requiere establecer estrategia clara de versionado, etiquetado de releases y gestión de ramas para mantener orden en repositorio.

Restricciones : Debe soportar múltiples versiones en paralelo, debe permitir hotfixes en versiones antiguas, debe mantener historial claro de cambios, debe integrar con proceso de publicación.

Producciones esperadas :

- Estrategia de versionado documentada (semantic versioning)
- Estructura de ramas definida (main, develop, release, hotfix)
- Procedimiento de creación de releases
- Archivo CHANGELOG actualizado
- Etiquetas de versión en repositorio
- Documentación de cómo hacer hotfix en versión anterior

Code Review y Colaboración en Repositorio

Contexto : El equipo de desarrollo implementa proceso de code review para mejorar calidad y compartir conocimiento. Se necesita establecer criterios de revisión, proceso de feedback y herramientas para facilitar colaboración.

Restricciones : El proceso debe ser eficiente (no bloquear desarrollo), debe asegurar calidad de código, debe facilitar aprendizaje entre desarrolladores, debe documentar decisiones técnicas.

Producciones esperadas :

- Guía de code review documentada
- Criterios de aceptación para pull requests
- Plantilla de pull request
- Documentación de cómo proporcionar feedback constructivo
- Métricas de tiempo de revisión
- Registro de decisiones técnicas tomadas en reviews

Pruebas esperadas

■ Pruebas documentales

- Archivo de configuración de pipeline CI/CD (YAML, JSON)
- Documentación de arquitectura en Markdown o formato similar
- Diagramas de arquitectura (C4, UML, flujos de datos)
- Archivo CHANGELOG con historial de cambios
- Documentación de procedimientos (compilación, despliegue, contribución)
- Reportes de análisis de calidad de código
- Documentación de decisiones arquitectónicas (ADR)
- Guía de code review y criterios de aceptación
- Configuración de herramientas de análisis estático
- Documentación de API interna

■ Pruebas de acción (en campo)

- Configurar pipeline CI/CD que ejecuta pruebas automáticamente
- Crear y mantener documentación de arquitectura actualizada
- Implementar análisis estático de código en pipeline
- Realizar code reviews siguiendo criterios establecidos
- Crear etiquetas de versión en repositorio
- Gestionar ramas de desarrollo siguiendo estrategia definida
- Resolver conflictos de merge en repositorio
- Publicar artefactos en repositorio de distribución
- Monitorear métricas de calidad de código
- Actualizar documentación cuando hay cambios arquitectónicos

■ Pruebas reflexivas

- Reflexión sobre efectividad del pipeline CI/CD y oportunidades de mejora
- Análisis de patrones de errores detectados por análisis estático
- Evaluación de claridad de documentación basada en feedback de nuevos desarrolladores
- Reflexión sobre procesos de code review y cómo mejorar colaboración
- Análisis de métricas de calidad y tendencias a lo largo del tiempo
- Evaluación de cómo decisiones arquitectónicas impactan mantenibilidad
- Reflexión sobre automatización de procesos y cuellos de botella
- Análisis de tiempo de resolución de problemas identificados por herramientas

Criterios de éxito (7 familias)

Familia	Definición	Ponderación
Conformité	Pipeline CI/CD ejecuta todas las pruebas automatizadas en cada commit sin excepciones	100% de commits tienen ejecución de pipeline completada
Qualité	Documentación de arquitectura es clara, completa y actualizada con cambios de código	Documentación cubre al menos 80% de componentes principales y se actualiza en cada release
Comunicación	Mensajes de commit son descriptivos y explican qué cambios se realizaron y por qué	100% de commits tienen mensajes de más de 10 palabras que explican cambios
Pertinence	Análisis de calidad de código identifica problemas reales que afectan mantenibilidad	Al menos 70% de problemas identificados por herramientas son corregidos o documentados
Traçabilité	Cada cambio en repositorio es trazable a requisito o issue específica	100% de pull requests referencian issue o requisito en descripción
Comunicación	Code reviews proporcionan feedback constructivo que mejora calidad y facilita aprendizaje	Feedback en reviews es específico, actionable y respetuoso en 100% de casos
Posture	Desarrollador mantiene disciplina en documentación y control de versiones de forma consistente	Documentación se actualiza en cada cambio significativo y commits siguen convenciones establecidas

Umbral de validación : Promedio ≥ 10/20

ANEXO VI Glosario y terminología

Este glosario presenta la terminología del referencial. Los términos siguientes deben utilizarse sistemáticamente en la enseñanza y la evaluación.

1. Términos profesionales obligatorios

- Abstracción de componentes
- Abstracción de lógica de red
- Accesibilidad en formularios
- Accesibilidad en interfaces móviles
- Actualización de dependencias
- Almacenamiento seguro de datos
- Almacenamiento seguro de tokens
- Análisis de cuota de mercado
- Análisis de errores
- Análisis de impacto en rendimiento
- Análisis de logs
- Análisis de rendimiento de plataformas
- Análisis de trazas de error
- Análisis de volcados de memoria
- Aplicación de metodologías UCD
- Aplicación de patrones MVC/MVP/MVVM
- Cierre de sesión por expiración
- Cobertura de pruebas
- Comparación de enfoques de desarrollo
- Conexión con dispositivos físicos
- Configuración de clientes HTTP
- Configuración de compilaciones de producción
- Configuración de emuladores y dispositivos virtuales
- Configuración de proyecto base
- Conocimiento de directrices de Apple
- Definición de alcance del producto
- Definición de escenarios de prueba
- Definición de métricas de rendimiento
- Desarrollo de Aplicaciones Móviles
- Desarrollo de componentes visuales
- Deserialización de JSON
- Diseño de arquitectura de información
- Diseño de esquema de datos
- Diseño de flujos de navegación
- Diseño de layouts responsivos
- Diseño de prototipos de alta fidelidad
- Diseño de prototipos de baja fidelidad
- Diseño de pruebas unitarias
- Documentación de alcance
- Documentación de API de componentes
- Documentación de arquitectura
- Documentación de cambios solicitados

- Documentación de casos de uso
- Documentación de decisiones técnicas
- Documentación de problemas de compatibilidad
- Ejecución de pruebas en matriz de dispositivos
- Especificación de criterios de aceptación
- Especificación técnica de funcionalidades
- Estimación de esfuerzo técnico
- Evaluación de accesibilidad
- Evaluación de desarrollo nativo
- Evaluación de frameworks multiplataforma
- Evaluación de mantenibilidad
- Evaluación de restricciones de dispositivos
- Evaluación de restricciones de dispositivos Apple
- Evaluación de restricciones técnicas móviles
- Evaluación de valor de negocio
- Evaluación del ecosistema Android
- Evaluación del ecosistema iOS
- Facilitación de comunicación entre equipos
- Generación de artefactos de distribución
- Gestión de cambios de alcance
- Gestión de claves de firma
- Gestión de códigos de respuesta HTTP
- Gestión de dependencias
- Gestión de errores de red
- Gestión de estado con ViewModel
- Gestión de JWT
- Gestión de migraciones de datos
- Gestión de paquetes con CocoaPods
- Gestión de paquetes con Gradle
- Gestión de permisos de notificaciones
- Gestión de respuestas incompletas
- Gestión del historial de navegación
- Gestión del teclado virtual
- Identificación de componentes reutilizables
- Identificación de dependencias necesarias
- Identificación de requisitos no funcionales
- Identificación de vulnerabilidades OWASP Mobile
- Implementación con Flutter Test
- Implementación con JUnit
- Implementación con XCTest
- Implementación de animaciones de transición
- Implementación de BLoC
- Implementación de cifrado en aplicaciones móviles
- Implementación de Core Data
- Implementación de flujos de autenticación
- Implementación de flujos de navegación
- Implementación de formularios
- Implementación de Human Interface Guidelines
- Implementación de llamadas a APIs
- Implementación de lógica de negocio
- Implementación de manejo de excepciones

- Implementación de Material Design
- Implementación de micro-interacciones
- Implementación de métodos HTTP
- Implementación de principios de usabilidad
- Implementación de Provider
- Implementación de pruebas con Appium
- Implementación de pruebas con Espresso
- Implementación de pruebas con XCUITest
- Implementación de recuperación de contraseña
- Implementación de Redux
- Implementación de repositorios
- Implementación de restricciones de layout
- Implementación de Room
- Implementación de SQLite
- Implementación de validaciones cliente
- Instalación y configuración de IDEs
- Integración con servicios Android
- Integración de Apple Push Notification service
- Integración de Firebase Auth
- Integración de Firebase Cloud Messaging
- Integración de OAuth
- Integración en pipeline CI/CD
- Iteración basada en feedback
- Iteración de diseño basada en usabilidad
- Justificación de decisiones tecnológicas
- Manejo de fallos de conectividad
- Manejo de notificaciones en primer plano
- Manejo de notificaciones en segundo plano
- Mantenimiento de biblioteca de componentes
- Mapeo de datos JSON a modelos
- Mensajes de error comprensibles
- Mensajes de error contextuales
- Modelado de datos
- Navegación entre campos de formulario
- Negociación de requisitos
- Obtención de aprobación formal
- Optimización de animaciones
- Optimización de flujos de estado
- Paso de parámetros entre pantallas
- Planificación de pruebas de compatibilidad
- Presentación de wireframes
- Priorización de backlog
- Priorización de defectos
- Programación orientada a objetos
- Protección de datos sensibles
- Prueba en múltiples tamaños de pantalla
- Pruebas de usabilidad
- Pruebas funcionales en Android
- Recopilación de feedback de stakeholders
- Recopilación de feedback de usuarios
- Recopilación de requisitos funcionales

- Redacción de historias de usuario
- Registro de errores (logging)
- Renovación automática de tokens
- Representación de flujos de interacción
- Reproducción de errores
- Resolución de conflictos de compatibilidad
- Revisión de seguridad de código
- Seguridad en autenticación
- Separación de responsabilidades
- Traducción de requisitos de negocio
- Trazabilidad de requisitos
- Uso de APIs de animación nativas
- Uso de componentes de navegación nativos
- Uso de herramientas de depuración
- Uso de herramientas de diseño (Figma, Sketch)
- Uso de unidades relativas
- Validación de compatibilidad multiplataforma
- Validación de datos recibidos
- Validación de entorno de desarrollo
- Validación de formato de datos
- Validación de patrones de interacción
- Validación de requisitos con stakeholders
- Validación servidor-cliente

2. Sinónimos normalizados (forma canónica en negrita)

- Alcance → **Definición de alcance del producto móvil**
- Animaciones → **Añadir animaciones e interacciones en la interfaz móvil**
- APIs → **Conectar la aplicación con APIs externas**
- Arquitectura → **Diseño de arquitectura de información de la aplicación**
- Autenticación → **Integración de autenticación de usuarios en la aplicación**
- Base de datos → **Implementar la persistencia local de datos en la aplicación**
- Compatibilidad → **Ejecutar pruebas de compatibilidad en distintos dispositivos**
- Compilación → **Preparar compilaciones para la publicación de la aplicación**
- Componentes → **Crear componentes reutilizables de interfaz**
- Dependencias → **Instalar dependencias y SDK necesarios para el proyecto**
- Depuración → **Depurar fallos y errores de la aplicación móvil**
- Entorno de desarrollo → **Configurar el entorno de desarrollo móvil**
- Formularios → **Desarrollar formularios de entrada de datos en la aplicación**
- Gestión de estado → **Gestión de estado de la aplicación móvil**
- Interfaz de usuario → **Interfaces de usuario móviles**
- JSON → **Procesar respuestas JSON de servicios remotos**
- Lógica de negocio → **Programar la lógica de negocio de la aplicación móvil**
- Manejo de errores → **Gestionar errores y excepciones de ejecución en la aplicación**
- Navegación → **Implementar la navegación entre pantallas de la aplicación**
- Notificaciones → **Implementación de notificaciones push en la aplicación**
- Persistencia de datos → **Implementación de persistencia local de datos**
- Plataforma → **Seleccionar la plataforma objetivo Android para el desarrollo**
- Priorización → **Priorizar historias de usuario para la entrega móvil**
- Prototipos → **Diseñar prototipos de pantallas móviles**
- Pruebas de compatibilidad → **Validación de compatibilidad multiplataforma**

- Pruebas de interfaz → **Desarrollar pruebas automatizadas de interfaz de usuario**
- Pruebas unitarias → **Escribir pruebas unitarias para la lógica de la aplicación**
- Requisitos funcionales → **Recopilación de requisitos funcionales**
- Requisitos no funcionales → **Identificación de requisitos no funcionales**
- Seguridad → **Aplicar principios de seguridad en el desarrollo de la aplicación**
- Servicios web → **Consumir servicios web REST desde la aplicación móvil**
- Validación → **Validar campos y entradas del usuario en la aplicación**
- Wireframes → **Validar wireframes con el equipo o clientes**

3. Glosario de términos

Término	Definición
Abstracción de componentes	Creación de componentes genéricos y configurables que encapsulan lógica y presentación para reutilización en múltiples contextos.
Abstracción de lógica de red	Encapsulación de detalles de comunicación de red en componentes reutilizables para simplificar resto de aplicación.
Accesibilidad en formularios	Aseguramiento de que formularios son usables por personas con discapacidades mediante etiquetas y navegación accesible.
Accesibilidad en interfaces móviles	Implementación de características que permiten a usuarios con discapacidades (visuales, auditivas, motoras) usar la aplicación efectivamente.
Actualización de dependencias	Proceso de actualizar librerías y frameworks a versiones más recientes, considerando compatibilidad y cambios de API.
Almacenamiento seguro de datos	Técnicas y prácticas para guardar información sensible en dispositivos móviles utilizando mecanismos de protección como cifrado, Keystore (Android) o Keychain (iOS).
Almacenamiento seguro de tokens	Guardado de tokens de autenticación en almacenamiento seguro del dispositivo (Keystore, Keychain) protegido de acceso no autorizado.
Análisis de cuota de mercado	Investigación de distribución de usuarios entre plataformas móviles (Android, iOS) para informar decisiones de desarrollo.
Análisis de errores	Examen de logs y reportes de errores para identificar patrones, causas raíz y oportunidades de mejora.
Análisis de impacto en rendimiento	Evaluación de cómo decisiones técnicas y arquitectónicas afectan la velocidad, eficiencia y experiencia de usuario de la aplicación.
Análisis de logs	Revisión de registros de eventos y errores para entender comportamiento de aplicación y diagnosticar problemas.
Análisis de rendimiento de plataformas	Evaluación de cómo diferentes plataformas y enfoques de desarrollo impactan en velocidad, consumo de recursos y experiencia del usuario.

Término	Definición
Análisis de trazas de error	Examen de stack traces para identificar ubicación exacta de errores y secuencia de llamadas que los causaron.
Análisis de volcados de memoria	Examen de volcados de memoria para identificar fugas de memoria y problemas de gestión de recursos.
Aplicación de metodologías UCD	Implementación de User-Centered Design, enfoque que coloca necesidades y comportamientos de usuarios en centro del diseño.
Aplicación de patrones MVC/MVP/MVVM	Uso de patrones arquitectónicos que separan la aplicación en capas (Modelo, Vista, Controlador/Presentador/ViewModel) para mejorar mantenibilidad.
Cierre de sesión por expiración	Implementación de cierre automático de sesión cuando token de autenticación expira, requiriendo nuevo login.
Cobertura de pruebas	Medida de porcentaje de código que es ejecutado por pruebas, indicando qué tan exhaustivas son las pruebas.
Comparación de enfoques de desarrollo	Evaluación de diferentes estrategias de desarrollo (nativo, multiplataforma, híbrido) considerando rendimiento, mantenibilidad y costo.
Conexión con dispositivos físicos	Configuración de conexión entre computadora de desarrollo y dispositivos móviles reales para pruebas y depuración.
Configuración de clientes HTTP	Instalación y configuración de librerías HTTP (Retrofit, Alamofire, Dio) para realizar solicitudes a servicios web.
Configuración de compilaciones de producción	Proceso de preparación de la aplicación para su distribución en entornos de producción, incluyendo optimizaciones, configuración de parámetros y eliminación de código de depuración.
Configuración de emuladores y dispositivos virtuales	Instalación y configuración de simuladores de dispositivos móviles para pruebas de desarrollo sin necesidad de hardware físico.
Configuración de proyecto base	Establecimiento de estructura inicial del proyecto, dependencias, configuraciones y convenciones de código para iniciar desarrollo.
Conocimiento de directrices de Apple	Comprensión de estándares, guías de diseño y requisitos técnicos establecidos por Apple para aplicaciones en el ecosistema iOS.
Definición de alcance del producto	Delimitación clara de funcionalidades, características y límites que incluye la aplicación móvil, diferenciando lo que está dentro y fuera del proyecto.
Definición de escenarios de prueba	Especificación de casos de prueba que validan funcionalidades críticas y flujos de usuario importantes.
Definición de métricas de rendimiento	Establecimiento de indicadores cuantificables (tiempo de respuesta, consumo de memoria, duración de batería) para evaluar el desempeño de la aplicación.

Término	Definición
Desarrollo de Aplicaciones Móviles	Proceso integral de creación, diseño, implementación y mantenimiento de aplicaciones de software diseñadas para ejecutarse en dispositivos móviles (smartphones, tablets) en plataformas como Android e iOS.
Desarrollo de componentes visuales	Creación de elementos de interfaz de usuario (botones, campos de texto, listas) que forman la presentación visual de la aplicación.
Deserialización de JSON	Conversión de datos en formato JSON recibidos de servicios web en objetos de aplicación.
Diseño de arquitectura de información	Estructuración lógica de cómo se organizan, relacionan y presentan los datos dentro de la aplicación móvil.
Diseño de esquema de datos	Definición de estructura de tablas, campos, relaciones e índices para base de datos local de la aplicación.
Diseño de flujos de navegación	Planificación de cómo los usuarios se desplazan entre pantallas y secciones de la aplicación, incluyendo transiciones y paso de datos.
Diseño de layouts responsivos	Creación de interfaces que se adaptan automáticamente a diferentes tamaños de pantalla y orientaciones manteniendo usabilidad.
Diseño de prototipos de alta fidelidad	Creación de prototipos detallados y realistas que simulan comportamiento y apariencia final de la aplicación.
Diseño de prototipos de baja fidelidad	Creación de representaciones simples y rápidas de conceptos de interfaz para validar ideas antes de desarrollo detallado.
Diseño de pruebas unitarias	Planificación de casos de prueba que validan funciones individuales de forma aislada.
Documentación de alcance	Registro formal y detallado de todas las funcionalidades, características y límites del producto móvil acordados con stakeholders.
Documentación de API de componentes	Registro detallado de propiedades, métodos, eventos y ejemplos de uso de componentes reutilizables para facilitar su adopción.
Documentación de arquitectura	Registro detallado de decisiones arquitectónicas, componentes, patrones y justificaciones técnicas de la aplicación móvil.
Documentación de cambios solicitados	Registro detallado de modificaciones solicitadas por stakeholders incluyendo justificación y impacto en proyecto.
Documentación de casos de uso	Descripción detallada de escenarios de interacción entre usuarios y la aplicación, incluyendo flujos normales y excepcionales.
Documentación de decisiones técnicas	Registro formal de decisiones arquitectónicas, de plataforma y tecnológicas con justificación y alternativas consideradas.

Término	Definición
Documentación de problemas de compatibilidad	Registro detallado de defectos, comportamientos inesperados y limitaciones encontradas durante pruebas en diferentes dispositivos y plataformas.
Ejecución de pruebas en matriz de dispositivos	Realización de pruebas funcionales y de compatibilidad en múltiples dispositivos físicos y emuladores con diferentes características y versiones de SO.
Especificación de criterios de aceptación	Definición clara de condiciones que deben cumplirse para que una funcionalidad o requisito sea considerado completado y aceptable.
Especificación técnica de funcionalidades	Descripción detallada de cómo se implementarán técnicamente las funcionalidades, incluyendo arquitectura, componentes y flujos de datos.
Estimación de esfuerzo técnico	Cálculo del tiempo, recursos y complejidad técnica requeridos para implementar una funcionalidad o historia de usuario.
Evaluación de accesibilidad	Análisis sistemático de si la aplicación es usable por personas con diferentes capacidades y discapacidades.
Evaluación de desarrollo nativo	Análisis de ventajas y desventajas de desarrollar aplicaciones específicamente para cada plataforma (Android o iOS) usando lenguajes nativos.
Evaluación de frameworks multiplataforma	Análisis de herramientas y frameworks (React Native, Flutter, Xamarin) que permiten desarrollar para múltiples plataformas con código compartido.
Evaluación de mantenibilidad	Análisis de facilidad para mantener, actualizar y extender la aplicación considerando código, documentación y arquitectura.
Evaluación de restricciones de dispositivos	Análisis de limitaciones técnicas específicas de dispositivos móviles (procesador, memoria, pantalla) que afectan el desarrollo.
Evaluación de restricciones de dispositivos Apple	Análisis de limitaciones técnicas específicas de dispositivos Apple (iPhone, iPad) que afectan el desarrollo de aplicaciones.
Evaluación de restricciones técnicas móviles	Análisis de limitaciones inherentes a dispositivos móviles como capacidad de procesamiento, memoria, batería y conectividad que afectan el diseño de la aplicación.
Evaluación de valor de negocio	Análisis del impacto y beneficio que cada funcionalidad aporta a los objetivos comerciales y satisfacción de usuarios.
Evaluación del ecosistema Android	Análisis de características, capacidades, limitaciones y oportunidades de la plataforma Android para desarrollo de aplicaciones.
Evaluación del ecosistema iOS	Análisis de características, capacidades, limitaciones y oportunidades de la plataforma iOS para desarrollo de aplicaciones.

Término	Definición
Facilitación de comunicación entre equipos	Actuación como intermediario para asegurar comprensión mutua y alineación entre equipos de negocio, diseño y desarrollo.
Generación de artefactos de distribución	Creación de paquetes ejecutables (APK para Android, IPA para iOS) listos para ser publicados en tiendas de aplicaciones o distribuidos a usuarios finales.
Gestión de cambios de alcance	Proceso controlado para evaluar, aprobar e implementar modificaciones al alcance original del proyecto, documentando impacto en tiempo y recursos.
Gestión de claves de firma	Administración segura de certificados digitales y claves criptográficas utilizadas para firmar aplicaciones móviles, garantizando autenticidad e integridad.
Gestión de códigos de respuesta HTTP	Procesamiento correcto de códigos de estado HTTP (200, 201, 400, 401, 404, 500) para determinar resultado de solicitud.
Gestión de dependencias	Identificación y administración de relaciones entre funcionalidades, tareas y componentes que deben completarse en orden específico.
Gestión de errores de red	Implementación de manejo robusto de errores de conectividad, timeouts y respuestas de error de servidores.
Gestión de estado con ViewModel	Implementación de patrón ViewModel que encapsula lógica de presentación y estado, separándolo de la vista.
Gestión de JWT	Manejo de JSON Web Tokens incluyendo validación, decodificación y renovación para autenticación basada en tokens.
Gestión de migraciones de datos	Proceso de actualizar esquema de base de datos entre versiones de aplicación sin pérdida de datos de usuarios.
Gestión de paquetes con CocoaPods	Uso de gestor de dependencias CocoaPods para descargar, actualizar y gestionar librerías en proyectos iOS.
Gestión de paquetes con Gradle	Uso de herramienta de construcción Gradle para descargar, actualizar y gestionar dependencias en proyectos Android.
Gestión de permisos de notificaciones	Solicitud y manejo de permisos del usuario para envío de notificaciones push en dispositivos móviles.
Gestión de respuestas incompletas	Manejo de situaciones donde respuesta de servidor no contiene todos los datos esperados o está parcialmente corrupta.
Gestión del historial de navegación	Implementación de mecanismo que permite a usuarios retroceder a pantallas anteriores manteniendo estado y contexto.
Gestión del teclado virtual	Control de aparición, desaparición y comportamiento del teclado virtual en relación con campos de entrada.

Término	Definición
Identificación de componentes reutilizables	Reconocimiento de elementos de interfaz que se repiten en múltiples pantallas y pueden ser abstraídos como componentes independientes.
Identificación de dependencias necesarias	Determinación de librerías, frameworks y componentes externos que el proyecto requiere para funcionar correctamente.
Identificación de requisitos no funcionales	Determinación de características técnicas y restricciones que la aplicación debe cumplir, como rendimiento, seguridad, escalabilidad y compatibilidad.
Identificación de vulnerabilidades OWASP Mobile	Análisis y detección de debilidades de seguridad en aplicaciones móviles según el estándar OWASP Mobile Top 10, incluyendo inyección, almacenamiento inseguro y criptografía débil.
Implementación con Flutter Test	Uso de framework de testing de Flutter para escribir pruebas unitarias en proyectos Flutter.
Implementación con JUnit	Uso de framework JUnit para escribir y ejecutar pruebas unitarias en proyectos Android.
Implementación con XCTest	Uso de framework XCTest para escribir y ejecutar pruebas unitarias en proyectos iOS.
Implementación de animaciones de transición	Codificación de efectos visuales suaves que acompañan cambios entre pantallas o estados de interfaz.
Implementación de BLoC	Aplicación de patrón BLoC (Business Logic Component) para separar lógica de negocio de presentación usando streams y eventos.
Implementación de cifrado en aplicaciones móviles	Aplicación de algoritmos criptográficos para proteger datos sensibles en tránsito y en reposo dentro de la aplicación móvil.
Implementación de Core Data	Uso de framework Core Data de iOS para gestión de persistencia de datos con mapeo objeto-relacional.
Implementación de flujos de autenticación	Codificación de procesos de login, registro y gestión de sesiones para autenticar usuarios en aplicación.
Implementación de flujos de navegación	Codificación de cómo los usuarios se desplazan entre pantallas, incluyendo transiciones, parámetros y gestión del historial.
Implementación de formularios	Creación de interfaces para recopilación de datos del usuario incluyendo campos, validación y envío.
Implementación de Human Interface Guidelines	Aplicación de directrices de diseño de Apple (HIG) en interfaces iOS, incluyendo componentes, navegación y patrones de interacción.
Implementación de llamadas a APIs	Codificación de solicitudes HTTP a servicios web remotos incluyendo parámetros, headers y manejo de respuestas.
Implementación de lógica de negocio	Codificación de reglas, procesos y algoritmos que implementan los requisitos funcionales de la aplicación.

Término	Definición
Implementación de manejo de excepciones	Codificación de mecanismos para capturar, procesar y recuperarse de errores durante ejecución de aplicación.
Implementación de Material Design	Aplicación de sistema de diseño Material Design de Google en interfaces Android, incluyendo componentes, colores y animaciones.
Implementación de micro-interacciones	Creación de pequeñas animaciones e interacciones que proporcionan feedback visual a acciones del usuario.
Implementación de métodos HTTP	Codificación de solicitudes HTTP usando métodos GET, POST, PUT, DELETE según operaciones requeridas.
Implementación de principios de usabilidad	Aplicación de principios de diseño que mejoran facilidad de uso, eficiencia y satisfacción del usuario con la aplicación.
Implementación de Provider	Uso de paquete Provider para gestión de estado en Flutter, proporcionando acceso a datos a través de InheritedWidget.
Implementación de pruebas con Appium	Uso de framework Appium para escribir pruebas automatizadas multiplataforma de interfaz de usuario.
Implementación de pruebas con Espresso	Uso de framework Espresso para escribir pruebas automatizadas de interfaz de usuario en Android.
Implementación de pruebas con XCUITest	Uso de framework XCUITest para escribir pruebas automatizadas de interfaz de usuario en iOS.
Implementación de recuperación de contraseña	Codificación de flujo que permite usuarios recuperar acceso a cuenta mediante verificación de identidad.
Implementación de Redux	Uso de patrón de gestión de estado Redux que centraliza estado de aplicación en un almacén único con acciones y reductores.
Implementación de repositorios	Creación de capa de abstracción que encapsula lógica de acceso a datos (local y remoto) para separar de lógica de negocio.
Implementación de restricciones de layout	Uso de sistemas de restricciones (ConstraintLayout, Auto Layout) para definir posiciones y tamaños de elementos de forma flexible.
Implementación de Room	Uso de librería Room de Android que proporciona capa de abstracción sobre SQLite con mapeo objeto-relacional.
Implementación de SQLite	Uso de base de datos SQLite para almacenamiento local de datos en aplicaciones Android, proporcionando persistencia relacional.
Implementación de validaciones cliente	Codificación de validaciones en dispositivo móvil que verifican datos antes de envío a servidor.
Instalación y configuración de IDEs	Descarga, instalación y configuración de entornos de desarrollo integrados (Android Studio, Xcode) para desarrollo de aplicaciones móviles.
Integración con servicios Android	Conexión e incorporación de servicios específicos del ecosistema Android (notificaciones, sensores, APIs de sistema) en la aplicación móvil.

Término	Definición
Integración de Apple Push Notification service	Uso de servicio APNs de Apple para envío de notificaciones push a dispositivos iOS.
Integración de Firebase Auth	Uso de servicio Firebase Authentication para gestión de autenticación de usuarios con múltiples métodos.
Integración de Firebase Cloud Messaging	Uso de servicio Firebase Cloud Messaging para envío de notificaciones push a dispositivos Android.
Integración de OAuth	Implementación de protocolo OAuth para permitir autenticación mediante proveedores externos (Google, Facebook, Apple).
Integración en pipeline CI/CD	Incorporación de pruebas automatizadas en proceso de integración continua para ejecución automática en cada compilación.
Iteración basada en feedback	Proceso de mejora continua donde cambios se implementan basándose en retroalimentación de usuarios y pruebas.
Iteración de diseño basada en usabilidad	Proceso de refinamiento de diseño basado en resultados de pruebas de usabilidad y feedback de usuarios.
Justificación de decisiones tecnológicas	Explicación fundamentada de por qué se seleccionaron tecnologías, plataformas y enfoques específicos para el proyecto.
Manejo de fallos de conectividad	Implementación de estrategias para aplicación funcione correctamente cuando hay pérdida de conexión de red.
Manejo de notificaciones en primer plano	Procesamiento de notificaciones push cuando aplicación está activa y visible al usuario.
Manejo de notificaciones en segundo plano	Procesamiento de notificaciones push cuando aplicación está cerrada o en segundo plano, incluyendo acciones silenciosas.
Mantenimiento de biblioteca de componentes	Gestión centralizada de componentes reutilizables incluyendo versionado, actualizaciones y control de calidad.
Mapeo de datos JSON a modelos	Transformación de estructura JSON en modelos de datos de aplicación, incluyendo conversión de tipos.
Mensajes de error comprensibles	Presentación de mensajes de error en lenguaje claro que explica qué salió mal y cómo solucionarlo.
Mensajes de error contextuales	Presentación de mensajes de error claros y específicos que ayudan usuarios a corregir problemas.
Modelado de datos	Definición de entidades, atributos, relaciones y restricciones que representan la información que maneja la aplicación.
Navegación entre campos de formulario	Implementación de flujo que permite usuarios moverse entre campos de formulario de forma eficiente.
Negociación de requisitos	Proceso de diálogo y acuerdo entre stakeholders para resolver conflictos, prioridades y restricciones en los requisitos de la aplicación.

Término	Definición
Obtención de aprobación formal	Proceso de conseguir consentimiento documentado de stakeholders clave antes de proceder a siguiente fase de desarrollo.
Optimización de animaciones	Mejora de rendimiento de animaciones para mantener fluidez (60 FPS) sin impacto en consumo de batería.
Optimización de flujos de estado	Mejora de eficiencia en gestión de estado reduciendo actualizaciones innecesarias y optimizando rendimiento de aplicación.
Paso de parámetros entre pantallas	Transmisión de datos entre diferentes pantallas de la aplicación para mantener contexto y permitir comunicación entre componentes.
Planificación de pruebas de compatibilidad	Definición estratégica de matriz de dispositivos, versiones de sistema operativo y configuraciones a probar para validar compatibilidad multiplataforma.
Presentación de wireframes	Comunicación de diseños de interfaz a stakeholders y equipo de desarrollo para obtener retroalimentación y aprobación.
Priorización de backlog	Ordenamiento de historias de usuario y funcionalidades en el backlog según criterios de valor de negocio, urgencia y dependencias técnicas.
Priorización de defectos	Clasificación y ordenamiento de errores encontrados según criterios como severidad, impacto en usuarios y esfuerzo de corrección para optimizar recursos de desarrollo.
Programación orientada a objetos	Paradigma de programación basado en objetos que encapsulan datos y comportamiento, promoviendo reutilización y mantenibilidad.
Protección de datos sensibles	Implementación de medidas para proteger información sensible (contraseñas, tokens, datos personales) de acceso no autorizado.
Prueba en múltiples tamaños de pantalla	Validación de que interfaces se ven y funcionan correctamente en diferentes tamaños de dispositivos (teléfono, tablet).
Pruebas de usabilidad	Evaluación de aplicación con usuarios reales para identificar problemas de usabilidad y oportunidades de mejora.
Pruebas funcionales en Android	Validación sistemática del comportamiento de la aplicación en la plataforma Android verificando que todas las funcionalidades se ejecutan según los requisitos especificados.
Recopilación de feedback de stakeholders	Obtención de opiniones y sugerencias de partes interesadas sobre diseños, funcionalidades y dirección del proyecto.
Recopilación de feedback de usuarios	Obtención sistemática de opiniones, sugerencias y críticas de usuarios reales sobre la aplicación para mejora continua.

Término	Definición
Recopilación de requisitos funcionales	Proceso de obtención y documentación de las funcionalidades específicas que la aplicación móvil debe implementar según necesidades de usuarios y negocio.
Redacción de historias de usuario	Formulación de requisitos funcionales en formato narrativo que describe qué hace un usuario, por qué lo hace y qué resultado espera obtener.
Registro de errores (logging)	Implementación de sistema que registra errores y eventos importantes para análisis y debugging.
Renovación automática de tokens	Implementación de mecanismo que renueva tokens de autenticación automáticamente antes de expiración.
Representación de flujos de interacción	Documentación visual de cómo usuarios interactúan con la aplicación, incluyendo transiciones y cambios de estado.
Reproducción de errores	Proceso de recrear condiciones que causan errores para entenderlos y validar soluciones.
Resolución de conflictos de compatibilidad	Identificación y solución de incompatibilidades entre versiones de dependencias que impiden compilación o funcionamiento correcto.
Revisión de seguridad de código	Análisis sistemático del código fuente de la aplicación para identificar vulnerabilidades, prácticas inseguras y posibles brechas de seguridad.
Seguridad en autenticación	Implementación de medidas de seguridad en procesos de autenticación como hash de contraseñas y protección contra ataques.
Separación de responsabilidades	Principio de diseño que asigna responsabilidades específicas a diferentes componentes para mejorar modularidad y mantenibilidad.
Traducción de requisitos de negocio	Conversión de necesidades y objetivos comerciales en especificaciones técnicas comprensibles para el equipo de desarrollo.
Trazabilidad de requisitos	Capacidad de rastrear cada requisito desde su origen en necesidades de negocio hasta su implementación y validación en la aplicación.
Uso de APIs de animación nativas	Utilización de APIs proporcionadas por plataformas (Android Animation, iOS Core Animation) para crear animaciones eficientes.
Uso de componentes de navegación nativos	Utilización de APIs y componentes proporcionados por plataformas (Android Navigation, iOS Navigation Controller) para gestionar navegación.
Uso de herramientas de depuración	Utilización de debuggers integrados en IDEs para inspeccionar estado de aplicación y ejecutar paso a paso.
Uso de herramientas de diseño (Figma, Sketch)	Utilización de software especializado para crear wireframes, prototipos y especificaciones visuales de interfaces móviles.

Término	Definición
Uso de unidades relativas	Utilización de medidas proporcionales (porcentajes, dp, sp) en lugar de píxeles fijos para crear interfaces adaptables.
Validación de compatibilidad multiplataforma	Proceso de verificación que asegura que la aplicación móvil funciona correctamente en múltiples plataformas (Android, iOS) y dispositivos con diferentes características técnicas.
Validación de datos recibidos	Verificación de que datos recibidos de servicios web cumplen formato, tipo y restricciones esperadas.
Validación de entorno de desarrollo	Verificación de que todas las herramientas, SDK y configuraciones necesarias están correctamente instaladas y funcionan adecuadamente.
Validación de formato de datos	Verificación de que datos ingresados cumplen formato esperado (email, teléfono, fecha, etc.).
Validación de patrones de interacción	Verificación de que patrones de interacción diseñados son intuitivos y efectivos para usuarios reales.
Validación de requisitos con stakeholders	Proceso de confirmación con partes interesadas (clientes, usuarios, gestores) de que los requisitos documentados son correctos, completos y alineados con objetivos.
Validación servidor-cliente	Implementación de validaciones tanto en cliente como en servidor para seguridad y consistencia de datos.