TEMA 5

Gestión de Proyectos Móviles

Programación de Dispositivos Móviles

Metodologías Tradicionales | Metodologías Ágiles | Selección de Plataforma | Seguimiento y Mantenimiento

CONTENIDOS

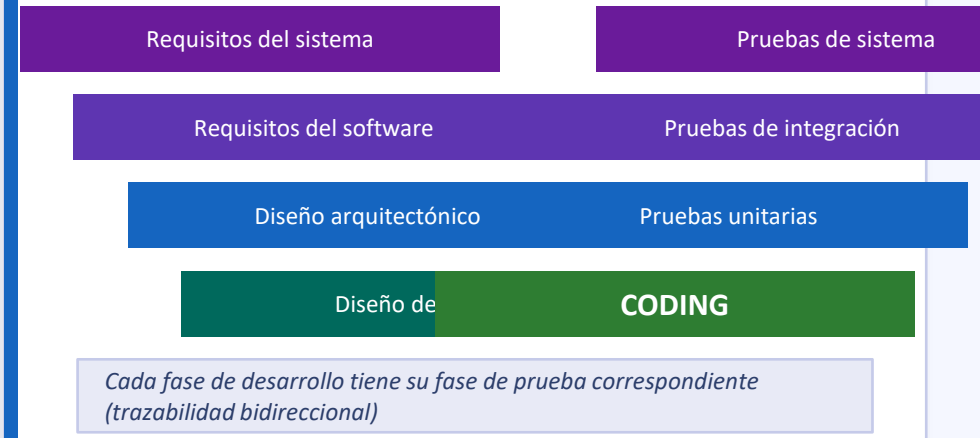
Tema 5

Waterfall y Modelo en V

Modelo en Cascada (Waterfall)

1. Requisitos
2. Diseño
3. Codificación
4. Pruebas
5. Despliegue
6. Mantenimiento

Modelo en V



Cada fase de desarrollo tiene su fase de prueba correspondiente (trazabilidad bidireccional)

Metodologías tradicionales: documentación exhaustiva, fases secuenciales, adecuadas para requisitos estables y proyectos con regulación estricta

Scrum: El Framework Ágil más Utilizado

Roles

PO — **Product Owner**
Define y prioriza el backlog

SM — **Scrum Master**
Facilita y elimina impedimentos

DT — **Dev Team**
Autoorganizado, 3-9 personas

Eventos (Sprint 2-4 semanas)

- **Sprint Planning**
  Planificación al inicio
- **Daily Scrum**
  15 min cada día (stand-up)
- **Sprint Review**
  Demo del incremento
- **Retrospectiva**
  Mejora del proceso

Artefactos

Product Backlog
Lista priorizada de funcionalidades (User Stories)

Sprint Backlog
Tareas comprometidas para el sprint actual

Incremento
Producto potencialmente entregable al final de cada sprint

Ciclo de un Sprint

1. **Product Backlog**
   User Stories priorizadas
2. **Sprint Planning**
   Seleccionar items del sprint
3. **Sprint Backlog**
   Tareas del sprint 1-4 sem.
4. **Daily Scrum**
   15 min · qué hice/haré/bloqueos
5. **Sprint Review**
   Demo + feedback stakeholders
6. **Retrospectiva**
   Mejoras para el próximo sprint

Product Backlog y User Stories

Formato de User Story

"Como [tipo de usuario],
quiero [funcionalidad]
para [beneficio/valor de negocio]"

Criterios de Aceptación (GIVEN-WHEN-THEN)

```
GIVEN [contexto inicial]
WHEN [acción del usuario]
THEN [resultado esperado verificable]
```

Ejemplo de Product Backlog — App de Gestión de Tareas

ID	User Story	Prioridad	Story Points	Sprint
US-01	Como usuario quiero registrarme con email/Google para acceder a mis tareas	Alta	8	1
US-02	Como usuario quiero crear, editar y eliminar tareas con fecha límite	Alta	5	1
US-03	Como usuario quiero organizar tareas en proyectos y categorías	Media	8	2
US-04	Como usuario quiero recibir notificaciones push antes de las fechas límite	Media	5	2
US-05	Como usuario quiero compartir proyectos y asignar tareas a otros miembros	Baja	13	3

Kanban y Herramientas de Gestión

Tablero Kanban

BACKLOG	EN PROGR WIP: 3	REVISIÓ WIP: 2	COMPLETADO
Perfil usuario	Registro	Crear tarea	Splash screen
Notif. push	Login Google	Lista tareas	Wireframes
Dark mode			

Principios Kanban: Visualizar el flujo · Limitar el WIP · Gestionar el flujo · Mejorar continuamente · Hacer políticas explícitas

Herramientas comparativa

Jira — *Equipos grandes*
- Scrum+Kanban
- Roadmaps
- Integ. CI/CD

Trello — *Equipos pequeños*
- Kanban visual
- Simple
- Gratis hasta 10 boards

Linear — *Dev teams*
- Muy rápido
- Ciclos
- Shortcuts

Notion — *Todo en uno*
- Docs+Tasks
- Flexible
- Bases de datos

iOS vs Android: Análisis para Tomar la Decisión

Aspecto	iOS (Apple)	Android (Google)
Cuota mercado global	~29%	~70%
Gasto promedio/usuario	2x más alto	Base
Fragmentación	Baja (~5 modelos activos)	Muy alta (1000+ modelos)
Ciclo de revisión	1-7 días (manual)	Horas (automático)
Lenguaje principal	Swift / SwiftUI	Kotlin / Jetpack Compose
Actualizaciones OS	~95% en 2 años	~50% en 2 años
Revenue share store	15-30%	15-30%
Ingresos por usuario	Mercados desarrollados	Mercados emergentes

Regla general: iOS primero si el público objetivo está en mercados desarrollados (Europa, EEUU, Japón) y tiene mayor poder adquisitivo. Android primero para máximo alcance global y mercados emergentes.

Estrategia Multiplataforma: Nativa vs Híbrida

MÁXIMO RENDIMIENTO

Nativa pura

Swift/Kotlin

+ Mejor rendimiento y UX
+ Acceso completo al hardware
+ Apple/Google guidelines
- Doble base de código
- Coste alto

EQUILIBRIO

React Native

JavaScript / TypeScript

+ ~80% código compartido
+ Componentes nativos reales
+ Ecosistema enorme (npm)
- Rendimiento inferior a nativa
- Bridge puede ser lento

TENDENCIA ACTUAL

Flutter

Dart

+ UI pixel-perfect consistente
+ Compilado a nativo (Skia/Impeller)
+ Hot reload rápido
- Lenguaje Dart menos común
- Tamaño del bundle mayor

LÓGICA COMPARTIDA

Kotlin Multiplatform

Kotlin

+ Comparte lógica de negocio
+ UI 100% nativa por plataforma
+ Interop con Swift/Obj-C
- UI duplicada aún necesaria
- Ecosistema más joven

Firebase Crashlytics: Monitoreo de Errores

Configuración e Implementación

```
// build.gradle.kts
implementation("com.google.firebase:firebase-
crashlytics-ktx")

// Crash manual (testing)
throw RuntimeException("Test crash!")

// Log personalizado antes del crash
FirebaseCrashlytics.getInstance().apply {
    setUserId(userId)
    setCustomKey("screen", "CheckoutActivity")
    setCustomKey("cart_items", cartSize)
    log("Usuario inició checkout")
}

// Reportar excepción no fatal
try {
    parseUserData(json)
} catch (e: JsonException) {
    FirebaseCrashlytics.getInstance()
        .recordException(e)
}
```

Dashboard: KPIs clave a vigilar

Crash-free users

Crash Rate

ANR Rate

Crashes by version

Affected users

Alertas automáticas:

• Nuevo issue detectado • Regresión de crash • Tasa supera umbral definido

Firebase Analytics: Métricas de Negocio

AD Adquisición

- DAU / MAU (usuarios activos)
- Fuentes de instalación
- Coste de adquisición (CAC)
- First open rate

AC Activación

- Tasa de onboarding completado
- Tiempo hasta primera acción clave
- Screens por sesión
- Feature adoption rate

RT Retención

- Retención D1 / D7 / D30
- Churn rate mensual
- Sesiones por usuario/semana
- DAU/MAU ratio

RV Ingresos (Revenue)

- ARPU (ingresos por usuario)
- LTV (valor de vida del usuario)
- Tasa de conversión premium
- MRR / ARR

RF Referral (viral)

- NPS (Net Promoter Score)
- Tasa de recomendación
- Shares / invitaciones
- K-factor viral

BH Comportamiento

- Funnel de conversión
- Pantallas más visitadas
- Eventos personalizados
- Tiempo en app por sesión

SLA, Ciclo de Actualizaciones y Soporte

SLA — Service Level Agreement

Prioridad	Tiempo respuesta	Tiempo resolución	Ejemplo
P1 - Crítico	1h	4h	App no arranca
P2 - Alto	4h	24h	Crash > 5% usuarios
P3 - Medio	24h	72h	Feature rota
P4 - Bajo	48h	1 sem	UI cosmética

Ciclo de actualizaciones

- **Hotfix** — Cuando sea necesario
- **Patch** — Cada 1-2 semanas
- **Feature** — Cada 4-6 semanas
- **Major** — Cada 3-6 meses

Comunicación de actualizaciones (Release Notes)

In-App

Modal al abrir la nueva versión: novedades + CTA

Push

Notificación: 'Versión 2.5 disponible — nuevas funciones'

Email

Newsletter a usuarios suscritos con changelog detallado

Play/App Store

Release notes obligatorias — afectan al rating y visibilidad

Resumen del Tema 5

Puntos Clave

01 Metodologías tradicionales (Waterfall, Modelo en V, Espiral) son adecuadas para requisitos estables y proyectos con regulación. Fases secuenciales con documentación exhaustiva.

02 Scrum (sprints 2-4 semanas) y Kanban (flujo continuo + WIP limits) son las metodologías ágiles más usadas. Herramientas: Jira, Trello, Linear, Notion.

03 iOS vs Android: iOS lidera en ingresos y mercados premium, Android en volumen global. Flutter y React Native son las mejores opciones multiplataforma.

04 Crashlytics monitorea la estabilidad (crash rate < 1%, ANR < 0.47%). Analytics mide adquisición, retención, ingresos y comportamiento del usuario.

05 El ciclo de mantenimiento incluye hotfixes urgentes, patches semanales y features cada 4-6 semanas. SLAs definen tiempos de respuesta por prioridad.

ACTIVIDADES PRÁCTICAS

Ejercicios del Tema 5

1 Comparar metodologías

Analizar un proyecto real y justificar si Waterfall o Scrum sería más adecuado según sus características.

2 Backlog completo

Definir 10 User Stories para una app de tu elección con criterios de aceptación GIVEN-WHEN-THEN.

3 Sprint Planning

Realizar un Sprint Planning de 2 semanas: seleccionar User Stories, asignar Story Points y crear el Sprint Backlog.

4 Tablero Kanban

Configurar un tablero Kanban en Trello o Linear con columnas, WIP limits y tarjetas para tu proyecto.

5 Decisión de plataforma

Crear una matriz de decisión iOS/Android/Multiplataforma para una startup con presupuesto limitado.

6 Crashlytics

Integrar Crashlytics en una app, generar un crash manual y analizar el informe en la consola Firebase.

7 Dashboard de métricas

Diseñar el dashboard de KPIs para tu app: qué medir, con qué herramienta y cuáles son los umbrales de alerta.