PROGRAMACIÓN DE DISPOSITIVOS MÓVILES

Tema 5

Gestión de Proyectos Móviles

> *Estos apuntes cubren las metodologías de gestión de proyectos aplicadas al desarrollo móvil: metodologías tradicionales (Waterfall, Modelo en V, Espiral), metodologías ágiles (Scrum, Kanban), criterios de selección de plataforma e idioma, y las herramientas de seguimiento y mantenimiento continuo (Crashlytics, Analytics, SLAs).*

Contenidos

1. Metodologías Tradicionales

Las metodologías tradicionales se caracterizan por su enfoque secuencial y estructurado, con fases bien definidas y documentación exhaustiva entre cada etapa. Son apropiadas cuando los requisitos son estables y conocidos desde el inicio.

1.1. Modelo en Cascada (Waterfall)

Descripción: Modelo de desarrollo lineal y secuencial donde cada fase debe completarse antes de iniciar la siguiente. Es el modelo más simple y el primero en formalizarse.

Fases del modelo

Fase	Descripción	Entregable principal
1. Requisitos	Captura y análisis de todos los requerimientos del sistema	Documento de especificación (SRS)
2. Diseño	Arquitectura del sistema, diseño de BD y UI	Documento de diseño técnico (SDD)
3. Codificación	Implementación del código basado en el diseño	Código fuente, APK/IPA
4. Pruebas	Verificación y validación del sistema completo	Plan de pruebas, informe de bugs
5. Despliegue	Publicación en tiendas y entrega al cliente	App publicada, manual de usuario
6. Mantenimiento	Corrección de bugs y actualizaciones menores	Parches, actualizaciones

Ventajas del Waterfall

- Proceso claro y bien documentado — fácil de gestionar y supervisar

- Hitos definidos que permiten medir el progreso de forma objetiva
- Adecuado para proyectos con requisitos completamente definidos desde el inicio
- Ideal para proyectos regulados que exigen documentación exhaustiva (médico, financiero, gubernamental)

Desventajas del Waterfall

- Muy difícil incorporar cambios una vez iniciado el desarrollo
- El cliente no ve el producto hasta el final del proceso
- Los errores de diseño detectados tarde son muy costosos de corregir
- En móviles, los requisitos suelen cambiar mucho durante el desarrollo

Cuándo usar Waterfall en proyectos móviles

- Aplicaciones para sectores regulados: salud (FDA/CE), banca, gobierno
- Proyectos con requisitos completamente cerrados y contratos de precio fijo
- Migraciones o integraciones de sistemas legacy con especificaciones claras
- Equipos distribuidos en múltiples países donde la comunicación continua es difícil

1.2. Modelo en V

Descripción: Extensión del modelo Waterfall que enfatiza la verificación y validación de cada fase de desarrollo. Cada fase de desarrollo tiene una fase de prueba correspondiente, estableciendo trazabilidad bidireccional.

- **Lado izquierdo (desarrollo):** Análisis → Diseño del sistema → Diseño arquitectónico → Diseño detallado → Codificación
- **Lado derecho (pruebas):** Pruebas unitarias → Pruebas de integración → Pruebas del sistema → Pruebas de aceptación
- **Principio clave:** Los planes de prueba se elaboran en paralelo con su fase de desarrollo correspondiente

> *Ventaja principal del Modelo en V: La trazabilidad bidireccional garantiza que cada requisito tiene al menos un caso de prueba que lo valida, reduciendo significativamente los defectos en producción.*

1.3. Modelo Espiral

Descripción: Modelo iterativo que combina el enfoque sistemático del Waterfall con la naturaleza iterativa de los prototipos. En cada iteración (espiral) se realiza planificación, análisis de riesgos, desarrollo y evaluación.

Cuatro cuadrantes por iteración:

1. **Determinación de objetivos:** Definir metas, alternativas y restricciones de la iteración
2. **Evaluación y reducción de riesgos:** Identificar y mitigar riesgos técnicos y de negocio

3. **Desarrollo y pruebas:** Implementar y verificar el prototipo o incremento
4. **Planificación:** Revisar resultados y planificar la próxima iteración

Aplicación en proyectos móviles:

- Apps complejas con alta incertidumbre técnica o de negocio
- Proyectos de gran escala con múltiples módulos interdependientes
- Cuando la gestión de riesgos es crítica (fintech, salud, infraestructuras)

2. Metodologías Ágiles

Las metodologías ágiles priorizan la entrega de valor al cliente de forma iterativa e incremental. Se basan en el **Manifiesto Ágil (2001)**: individuos e interacciones sobre procesos y herramientas, software funcionando sobre documentación exhaustiva, colaboración con el cliente sobre negociación contractual, y responder al cambio sobre seguir un plan.

2.1. Scrum

Descripción: Framework ágil que organiza el trabajo en sprints de 2-4 semanas, con reuniones diarias, revisiones y retrospectivas. Es el método ágil más popular en el desarrollo de apps móviles.

Roles en Scrum

Rol	Responsabilidad principal	Habilidades clave
Product Owner (PO)	Define y prioriza el Product Backlog. Representa al cliente.	Visión de negocio, comunicación, toma de decisiones
Scrum Master (SM)	Facilita el proceso Scrum y elimina impedimentos del equipo.	Coaching, resolución de conflictos, conocimiento Scrum
Dev Team	Autoorganizado, entrega el incremento en cada sprint. 3-9 personas.	Desarrollo, testing, diseño, DevOps — multidisciplinar

Eventos de Scrum

Evento	Duración	Objetivo
Sprint Planning	2-4 horas	Seleccionar ítems del backlog y planificar el sprint
Daily Scrum	15 minutos	Sincronización diaria: qué hice, qué haré, qué bloquea
Sprint Review	1-2 horas	Demo del incremento al PO y stakeholders, feedback
Retrospectiva	1-2 horas	Mejorar el proceso del equipo para el próximo sprint

Artefactos de Scrum

- **Product Backlog:** Lista priorizada de todas las funcionalidades, mejoras y correcciones pendientes. Ordenada por valor de negocio. El PO es su responsable.
- **Sprint Backlog:** Subconjunto del Product Backlog seleccionado para el sprint actual, más el plan del equipo para entregarlo.
- **Incremento:** Suma de todos los ítems del Product Backlog completados durante el sprint. Debe estar en estado potencialmente entregable.

Product Backlog y User Stories

Formato de User Story:

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

Criterios de Aceptación — formato GIVEN-WHEN-THEN:

```
GIVEN [contexto inicial del sistema o usuario]
WHEN  [acción específica que realiza el usuario]
THEN  [resultado esperado y verificable]
```

```
Ejemplo:
GIVEN el usuario no ha iniciado sesión
WHEN  introduce email y contraseña correctos y toca 'Entrar'
THEN  la app navega a la pantalla principal y muestra su nombre
```

Story Points (estimación relativa):

- Unidad abstracta que mide complejidad, no tiempo
- Escala de Fibonacci: 1, 2, 3, 5, 8, 13, 21 (no usar números mayores)
- **Técnica Planning Poker:** Todos los miembros votan simultáneamente → consenso → estimación
- **Velocity:** Story Points completados por sprint. Se estabiliza en 3-5 sprints

Burndown Chart

El Burndown Chart muestra el trabajo restante (story points) a lo largo del tiempo del sprint. Permite detectar desviaciones antes de que sea demasiado tarde:

- **Línea ideal:** Descenso lineal desde los story points comprometidos hasta 0 al final del sprint
- **Por encima de la línea ideal:** El equipo va retrasado — necesita revisar el scope
- **Por debajo de la línea ideal:** El equipo va adelantado — puede añadir más ítems

2.2. Kanban

Descripción: Sistema de gestión del flujo de trabajo basado en la visualización y limitación del trabajo en progreso (WIP). No tiene sprints fijos — el trabajo fluye continuamente de izquierda a derecha en el tablero.

Tablero Kanban

Columna	WIP Limit	Qué contiene	Criterio de avance
Backlog / TODO	Sin límite	Ítems por hacer, priorizados	Disponible para empezar
En Progreso / DOING	2-3 máx.	Tareas activamente en desarrollo	Código listo para review
En Revisión / REVIEW	2 máx.	Code review o QA en curso	Aprobado sin comentarios
Completado / DONE	Sin límite	Tareas finalizadas y validadas	—

Principios de Kanban

5. **Visualizar el flujo:** Todo el trabajo visible en el tablero — nada oculto
6. **Limitar el WIP:** Cuantas más tareas simultáneas, más lento el flujo — menos es más
7. **Gestionar el flujo:** Optimizar el ciclo de vida de cada tarea de principio a fin
8. **Hacer políticas explícitas:** Definir claramente los criterios de entrada/salida de cada columna
9. **Mejora continua:** Usar métricas para identificar cuellos de botella y mejorar

Métricas Kanban

- **Lead Time:** Tiempo desde que se crea una tarea hasta que se entrega. Mide la eficiencia total del proceso.
- **Cycle Time:** Tiempo desde que se empieza a trabajar una tarea hasta que termina. Mide la velocidad del equipo.
- **Throughput:** Número de tareas completadas por unidad de tiempo. Mide la productividad real.
- **CFD (Cumulative Flow Diagram):** Muestra la acumulación de tareas en cada columna a lo largo del tiempo. Detecta cuellos de botella visualmente.

2.3. Herramientas de Gestión

Herramienta	Metodologías	Precio base	Ideal para	Integración CI/CD
Jira	Scrum + Kanban	Gratis hasta 10 users	Equipos medianos/grandes	GitHub, GitLab, Jenkins
Trello	Kanban	Gratis (básico)	Equipos pequeños	Power-Ups limitados
Linear	Scrum + Kanban	Gratis (equipo)	Dev teams de startups	GitHub, GitLab, Vercel
Notion	Flexible	Gratis (personal)	Todo en uno (docs+tasks)	Limitada (Zapier)
Azure DevOps	Scrum + Kanban	Gratis hasta 5 users	Empresas grandes	Azure Pipelines nativo

3. Selección de Plataforma y Público

3.1. iOS vs Android: Análisis Técnico

Dimensión	iOS (Apple)	Android (Google)
Cuota de mercado global	~29% (dominante en EE.UU., Japón, Australia)	~70% (dominante en Europa, Asia, LATAM)
Gasto por usuario	2x mayor que Android en media global	Base de referencia
Fragmentación de dispositivos	Baja (~5 tamaños de pantalla activos)	Muy alta (1.000+ modelos distintos)
Ciclo de revisión	1-7 días (revisión manual por Apple)	Horas (mayoritariamente automático)
Lenguaje principal	Swift / SwiftUI (Obj-C heredado)	Kotlin / Jetpack Compose (Java heredado)
Actualizaciones del OS	~95% actualizan en 2 años	~50% actualizan en 2 años
Revenue share store	15% (ingresos <$1M) o 30%	15% (ingresos <$1M) o 30%
Herramientas de desarrollo	Xcode (solo macOS) — requiere Mac	Android Studio (Windows, macOS, Linux)

> *Regla general: iOS primero si el público está en mercados desarrollados con alto poder adquisitivo (Europa, EE.UU., Japón). Android primero para máximo alcance global y mercados emergentes (LATAM, África, Asia del Sur).*

3.2. Estrategias de Selección

Matriz de decisión para seleccionar plataforma

Factor	Solo iOS	Solo Android	Multiplataforma
Presupuesto limitado	Si el target es premium	Si el target es masivo	Mejor opción
Target: alto poder adquisitivo	Ideal	Funciona	Viable
Mercado global y diverso	No recomendado	Bueno	Excelente
App con funciones de hardware avanzadas	Muy bueno	Muy bueno	Limitado
Time-to-market urgente	Puede ser	Puede ser	React Native/Flutter

3.3. Estrategia Multiplataforma

Framework	Lenguaje	UI	Rendimiento	Cuándo elegirlo
Nativa pura	Swift/Kotlin	100% nativa	Máximo	Sin restricciones de presupuesto, máxima calidad
React Native	JavaScript/TS	Componentes nativos	Bueno	Equipo web, ecosistema npm, gran comunidad
Flutter	Dart	Widgets propios (Skia)	Muy bueno	UI consistente, compilado a nativo, soporte Google

Framework	Lenguaje	UI	Rendimiento	Cuándo elegirlo
KMP (Kotlin Multiplatform)	Kotlin	Nativa por plataforma	Nativo	Compartir solo lógica, UI nativa en cada plataforma

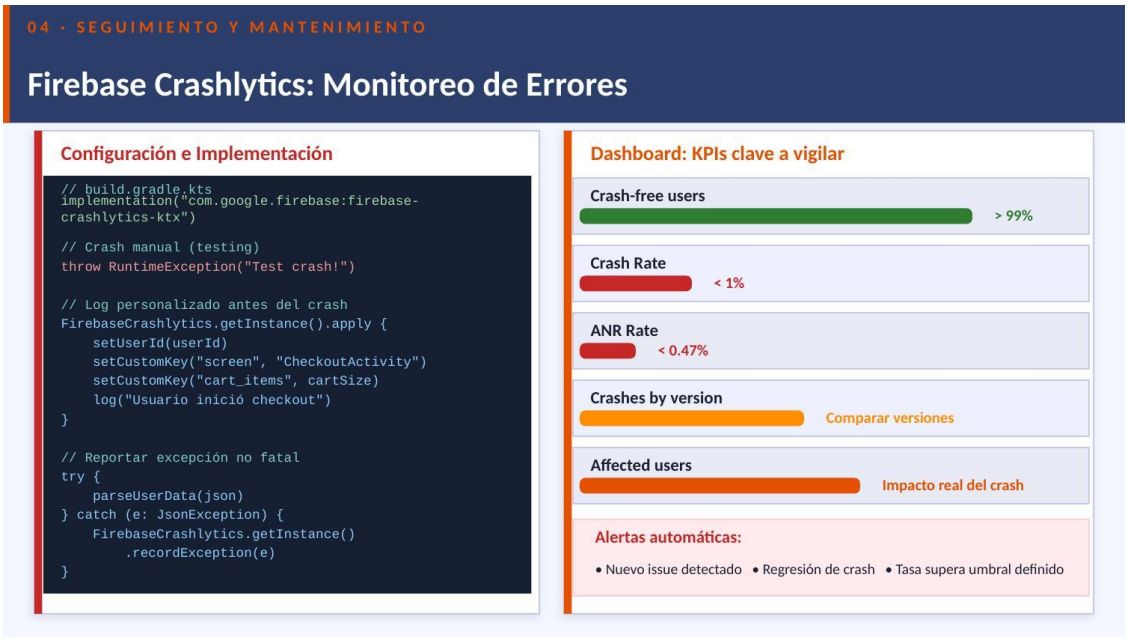
Flutter: La opción más popular actualmente

- **Lenguaje Dart:** Curva de aprendizaje moderada, similar a Java/Kotlin
- **Hot reload:** Ver cambios en milisegundos sin recompilar toda la app
- **Widget tree:** UI completamente controlada por Flutter (independiente del OS)
- **Soporte multiplataforma:** iOS, Android, Web, Windows, macOS, Linux desde una sola base de código
- **Rendimiento:** Compila a código nativo ARM — rendimiento muy cercano al nativo puro

React Native: El camino para equipos web

- **JavaScript/TypeScript:** Equipo web puede desarrollar apps móviles sin aprender Kotlin/Swift
- **Componentes nativos reales:** Renderiza componentes UI nativos del SO, no un WebView
- **Ecosistema npm:** Acceso a miles de librerías del ecosistema JavaScript
- **Mantenido por Meta:** Base de usuarios muy grande, comunidad activa y madura

4. Seguimiento y Mantenimiento



4.1. Monitoreo de Errores con Firebase Crashlytics

Firebase Crashlytics es la herramienta de Google para monitoreo de crashes en tiempo real. Proporciona informes detallados con stack traces, información del dispositivo y estado de la app en el momento del crash.

Implementación básica

```
// 1. Dependencia (build.gradle.kts)
implementation("com.google.firebase:firebase-crashlytics-ktx")

// 2. Añadir contexto personalizado antes de un posible crash
FirebaseCrashlytics.getInstance().apply {
    setUserId(currentUser.uid)
    setCustomKey("screen", "CheckoutActivity")
    setCustomKey("cart_items", cartSize)
    log("Usuario pulsó 'Confirmar pedido'")
}

// 3. Reportar excepciones no fatales (no crashan la app pero son
importantes)
try {
    val data = parseServerResponse(json)
} catch (e: JsonParseException) {
    FirebaseCrashlytics.getInstance().recordException(e)
    showFallbackUI()
}

// 4. Probar en desarrollo (crear crash manual)
throw RuntimeException("Test crash para verificar Crashlytics")
```

KPIs de estabilidad a monitorear

Métrica	Umbral objetivo	Herramienta	Acción si se supera
Crash-free users	> 99%	Crashlytics Dashboard	Hotfix urgente + rollback si es necesario
Crash Rate	< 1%	Crashlytics Dashboard	Pausar rollout, investigar causa raíz
ANR Rate	< 0.47%	Google Play Console	Optimizar operaciones en el hilo principal
Crashes by OS version	Detectar regresiones	Crashlytics + filtros	Desactivar feature con Remote Config
Affected users por issue	Priorizar por impacto	Crashlytics Issues	Resolver primero los que afectan a más usuarios

Alertas automáticas configurables

- **Nuevo issue detectado:** Notificación inmediata cuando aparece un crash que no existía antes
- **Regresión de crash:** Un issue previamente cerrado vuelve a aparecer en producción
- **Umbral de velocidad superado:** La tasa de crash supera el % definido en un periodo de tiempo
- **Canales de alerta:** Email, Slack (webhook), PagerDuty, alertas de Google Cloud

4.2. Análisis de Métricas con Firebase Analytics

Implementación de eventos personalizados

```
// Implementación básica de Analytics
val analytics = FirebaseAnalytics.getInstance(this)
```

```
// Evento personalizado con parámetros
val params = Bundle().apply {
    putString("item_id", product.id)
    putString("item_name", product.name)
    putDouble("price", product.price)
    putString("currency", "EUR")
}
analytics.logEvent(FirebaseAnalytics.Event.ADD_TO_CART, params)

// Evento de compra completada
analytics.logEvent(FirebaseAnalytics.Event.PURCHASE, Bundle().apply {
    putString(FirebaseAnalytics.Param.TRANSACTION_ID, orderId)
    putDouble(FirebaseAnalytics.Param.VALUE, orderTotal)
    putString(FirebaseAnalytics.Param.CURRENCY, "EUR")
})

// Propiedad de usuario (segmentación)
analytics.setUserProperty("subscription_type", "premium")
```

Métricas clave por dimensión

Dimensión	Métricas principales	Objetivo típico
Adquisición	DAU, MAU, fuentes de instalación, CAC	Entender de dónde vienen los usuarios
Activación	Tasa de onboarding, tiempo hasta acción clave	Reducir fricción en los primeros 5 minutos
Retención	D1/D7/D30 retention, churn rate, DAU/MAU	D1 > 35%, D7 > 15%, D30 > 5%
Ingresos	ARPU, LTV, tasa de conversión, MRR	LTV ≥ 3x CAC para sostenibilidad
Referral	NPS, K-factor viral, shares e invitaciones	NPS > 50 es excelente
Comportamiento	Funnel de conversión, pantallas más vistas	Detectar donde abandonan los usuarios

4.3. Soporte y Actualizaciones Continuas

SLA (Service Level Agreement)

El SLA define contractualmente los tiempos máximos de respuesta y resolución para cada nivel de prioridad de incidencia:

Prioridad	T. Respuesta	T. Resolución	Ejemplo típico	Acción
P1 — Crítico	1 hora	4 horas	App no arranca, datos perdidos	Hotfix inmediato
P2 — Alto	4 horas	24 horas	Crash en flujo principal > 5%	Hotfix en 24h
P3 — Medio	24 horas	72 horas	Feature rota para algunos usuarios	Parche semanal
P4 — Bajo	48 horas	1 semana	Problema cosmético o de texto	Próxima release

Ciclo de actualizaciones

Tipo de release	Frecuencia	Qué incluye	Proceso
Hotfix	Cuando sea necesario	Corrección crítica única	PR urgente → review rápida → deploy directo
Patch (x.x.X)	Cada 1-2 semanas	2-5 bugs no críticos	Sprint bugfix → QA básico → staged rollout
Feature Release (x.X.0)	Cada 4-6 semanas	Nuevas funcionalidades	Sprint completo → QA → beta → staged rollout
Major Release (X.0.0)	Cada 3-6 meses	Rediseño o nueva arquitectura	Múltiples sprints → beta pública → rollout lento

Release Notes — Buenas prácticas

- Usar lenguaje claro y comprensible para el usuario final, no jerga técnica
- **Estructura sugerida:** 🆕 Novedades · 🩹 Correcciones · ⚡ Mejoras de rendimiento
- Las Release Notes afectan directamente al rating en la store — son leídas por los usuarios
- Personalizar para el idioma y mercado: notas en español para España/LATAM, en inglés para el resto

```
Versión 2.5.0 — Enero 2025

🆕 NOVEDADES
• Modo oscuro: ahora disponible en Ajustes > Apariencia
• Sincronización con Google Calendar para gestión de fechas
• Filtros avanzados por etiqueta, prioridad y asignado

🩹 CORRECCIONES
• Solucionado el problema de notificaciones duplicadas en Android 14
• Corregido el crash al adjuntar archivos de más de 50 MB

⚡ MEJORAS
• La carga inicial es un 40% más rápida
• Menor consumo de batería en segundo plano
```

5. Resumen y Actividades

5.1. Puntos Clave del Tema

> *Metodologías tradicionales (Waterfall, Modelo en V, Espiral) son adecuadas para proyectos con requisitos estables, regulación estricta o equipos distribuidos. Implican documentación exhaustiva y fases secuenciales sin posibilidad de cambio fácil.*

> *Scrum organiza el trabajo en sprints de 2-4 semanas con 3 roles (PO, Scrum Master, Dev Team), 4 eventos (Planning, Daily, Review, Retrospectiva) y 3 artefactos (Product Backlog, Sprint Backlog, Incremento). Kanban usa un tablero visual con WIP limits para gestionar el flujo continuo.*

> *iOS lidera en ingresos y mercados premium (EE.UU., Europa, Japón). Android domina en volumen global. Flutter (Dart) y React Native (JavaScript) son las mejores opciones multiplataforma según el equipo disponible y los requisitos de rendimiento.*

> *Firebase Crashlytics monitorea la estabilidad: crash rate objetivo < 1%, ANR < 0.47%. Firebase Analytics mide 6 dimensiones: adquisición, activación, retención, ingresos, referral y comportamiento. Los SLAs definen tiempos de respuesta por prioridad de incidencia.*

> *El ciclo de mantenimiento incluye: hotfixes urgentes cuando sea necesario, patches semanales (bugs menores), feature releases cada 4-6 semanas y major releases cada 3-6 meses. Las release notes en lenguaje claro y orientadas al usuario impactan directamente en el rating de la app.*

5.2. Actividades Prácticas

10. **Comparativa de metodologías:** Analiza un proyecto de desarrollo de app real (o propuesto) y justifica razonadamente si Waterfall, Scrum o Kanban sería más adecuado, explicando los criterios de decisión utilizados.
11. **Product Backlog completo:** Define 10 User Stories para una app de tu elección. Cada historia debe incluir el formato 'Como... quiero... para...' y al menos 2 criterios de aceptación en formato GIVEN-WHEN-THEN.
12. **Sprint Planning simulado:** Selecciona 5 User Stories del backlog anterior, asigna Story Points usando Planning Poker, establece el Sprint Goal y elabora el Sprint Backlog con las tareas técnicas necesarias.
13. **Tablero Kanban:** Configura un tablero Kanban en Trello, Linear o Jira para tu proyecto. Crea columnas con sus WIP limits, añade 10 tarjetas reales y experimenta con el flujo durante una semana.
14. **Decisión de plataforma:** Crea una matriz de decisión completa para una startup con presupuesto de €30.000 que quiere lanzar una app de fitness. Justifica si elegir iOS, Android o una solución multiplataforma.
15. **Integración Crashlytics:** Integra Firebase Crashlytics en una app Android existente. Genera un crash manual con throw RuntimeException, localiza el informe en la consola y documenta la información que ofrece.
16. **Dashboard de KPIs:** Diseña el dashboard de métricas para tu app: qué 10 KPIs medirías, con qué herramienta, cuáles serían los umbrales de alerta y qué acción tomarías si cada uno se incumple.

5.3. Recursos Adicionales

- Scrum Guide — Guía oficial de Scrum (gratis en español)
- Atlassian Agile Coach — Guías de Scrum y Kanban
- Firebase Analytics Documentation — Documentación oficial
- Firebase Crashlytics Documentation — Implementación y dashboard
- Lean Startup Methodology — El método Lean aplicado a startups
- Kotlin Multiplatform — Documentación oficial de JetBrains