PLAN DE PRUEBAS DE SOFTWARE

ISO/IEC/IEEE 29119-3:2013

MinGO

Aplicación Móvil para el Aprendizaje de LSEC

Versión 1.0

Fecha: 19 de enero de 2026

Control del Documento

Título	Plan de Pruebas de Software - MinGO
Identificador	MINGO-TP-001-v1.0
Versión	1.0
Fecha	19/1/2026
Autor	Mateo Sosa, Fernando Tipán
Aprobado por	Marcos Escobar
Estándar	ISO/IEC/IEEE 29119-3:2013

Historial de Revisiones

Versión	Fecha	Autor	Cambios
1.0	19/1/2026	Marcos Escobar, Mateo Sosa, Fernando Tipán	Versión inicial. 81 casos de prueba totales (18 unitarias + 63 integración)

1. Introducción

1.1 Propósito

Este documento define el Plan de Pruebas para MinGO, estableciendo estrategia, alcance, recursos, cronograma y criterios para garantizar la calidad del software según ISO/IEC/IEEE 29119-3:2013.

1.2 Alcance

Tipos de prueba incluidos:

- Pruebas Unitarias: 18 test suites con Jest
- Pruebas de Integración: 63 casos con Postman
- Pruebas de Sistema: Flujos end-to-end críticos
- Pruebas de Regresión: Automatizadas en CI/CD

Excluido de este plan:

- Pruebas de carga/performance (plan separado)
- Pruebas de seguridad penetración (especialistas externos)
- Pruebas de usabilidad formal (fase posterior)

1.3 Referencias

- ISO/IEC/IEEE 29119-3:2013 - Software Testing Documentation
- MinGO SRS v2.0
- MinGO Design Document v1.0

2. Identificación del Software a Probar

Producto	MinGO - App educativa LSEC
Versión	v1.0.0
Backend	NestJS + TypeScript + Prisma ORM
Frontend	Flutter + Dart + BLoC Pattern
Base de Datos	PostgreSQL 15 (Supabase)
Arquitectura	Clean Architecture + DDD

2.1 Módulos a Probar

Módulo	Componentes	Casos de Prueba
Auth	Register, Login, Verify, Reset Password	20 (Postman) + 4 (Unit)
Children	CRUD hijos	12 (Postman) + 4 (Unit)
Progress	Tracking, Attempts, Completion	8 (Postman)
Content	Levels, Modules, Lessons	8 (Postman)
Classes	Create, Join, Assignments	15 (Postman)
Value Objects	Email, Password, UUID	3 (Unit)
Entities	User, ChildInfo	2 (Unit)
Services	JWT, Email, Password Hash	3 (Unit)
Controllers	AuthController	1 (Unit)

Total: 81 casos de prueba

3. Objetivos del Plan de Pruebas

Objetivos principales:

- Verificar que MinGO cumple con todos los requisitos funcionales del SRS v1.0
- Validar la correcta integración entre Backend API y Frontend móvil
- Garantizar la seguridad de autenticación y autorización (JWT, roles)
- Asegurar la integridad de datos en operaciones CRUD
- Detectar y corregir defectos antes del release v1.0
- Alcanzar cobertura de código >= 80% en componentes críticos
- Validar flujos de usuario end-to-end sin errores bloqueantes

3.1 Objetivos Específicos por Tipo de Prueba

Tipo de Prueba	Objetivo Específico
Unitarias	Verificar lógica de negocio aislada en Use Cases, Value Objects, Entities y Services. Target: 80% coverage.
Integración	Validar comunicación correcta entre capas (Controllers → Use Cases → Repositories) y respuestas HTTP de API REST.
Sistema	Comprobar flujos completos: Registro → Login → Navegación de contenido → Completar lección → Ver progreso.
Regresión	Asegurar que nuevos cambios no rompen funcionalidad existente. Ejecutar suite completa en cada PR.

4. Alcance de las Pruebas

4.1 En Alcance

Lo que SÍ se probará:

- Backend API: Todos los endpoints REST documentados (Auth, Children, Progress, Content, Classes)
- Autenticación: JWT tokens, refresh, roles (PADRE/DOCENTE), guards
- Validación de datos: DTOs, Value Objects, reglas de negocio
- Persistencia: Operaciones CRUD en PostgreSQL via Prisma

- Lógica de dominio: Use Cases, Entities, reglas (ej: 80% accuracy para completar)
- Gestión de errores: Códigos HTTP correctos, mensajes descriptivos
- Integración API-BD: Transacciones, integridad referencial

4.2 Fuera de Alcance

Lo que NO se probará en esta versión:

- UI/UX Flutter: Requiere pruebas manuales o E2E con framework específico
- Hand tracking MediaPipe: Necesita dispositivos físicos, no disponibles en CI
- Pasarela de pago Stripe: Se usará modo sandbox en fase posterior
- Notificaciones push Firebase: Validación en release posterior
- Performance bajo carga: Plan de pruebas de carga separado
- Seguridad penetración: Requiere auditores externos especializados

4.3 Funcionalidades Críticas (Prioridad Alta)

- Autenticación y autorización (RegisterUseCase, LoginUseCase, JWT)
- Gestión de hijos (CreateChildUseCase, validación edad 3-12)
- Progreso de aprendizaje (RecordAttemptUseCase, cálculo accuracy)
- Navegación de contenido (GetLevelsUseCase, desbloqueo de niveles)
- Clases virtuales (CreateClassUseCase, JoinClassUseCase, código único)

5. Estrategia de Pruebas

5.1 Tipos de Pruebas a Aplicar

Tipo	Herramienta	Responsable	Frecuencia
Unitarias	Jest	Developers	Cada commit
Integración	Postman + Newman	QA + Devs	Cada PR
Sistema	Manual + Postman	QA Team	Cada sprint
Regresión	Jest + Newman	CI/CD	Cada merge a develop
Aceptación	Manual	Product Owner	Pre-release

5.2 Niveles de Pruebas

Nivel 1: Pruebas Unitarias (18 test suites)

Verifican componentes aislados: Use Cases, Value Objects, Entities, Services. Usan mocks para dependencias externas. Ejecutadas con Jest en cada commit.

Nivel 2: Pruebas de Integración (63 casos)

Validan interacción entre componentes: Controllers → Use Cases → Repositories → BD. Ejecutadas con Postman/Newman contra API REST desplegada. Incluyen validación de contratos HTTP.

Nivel 3: Pruebas de Sistema (Flujos E2E)

Comprueban flujos completos de usuario: Registro → Login → Contenido → Progreso. Ejecutadas manualmente por QA o con colección Postman secuencial.

5.3 Técnicas de Prueba

- Caja Negra: Validación de entradas/salidas sin conocer implementación interna (Postman)
- Caja Blanca: Verificación de lógica interna con acceso al código (Jest unit tests)
- Partición de equivalencia: Agrupar inputs en clases válidas/inválidas
- Valores límite: Probar bordes (edad 3-12: probar 2, 3, 12, 13)
- Pruebas de estado: Validar transiciones (usuario no verificado → verificado → activo)
- Pruebas negativas: Forzar errores para validar manejo (401, 403, 404, 409, 500)

5.4 Automatización

80% de pruebas automatizadas (unitarias + integración). CI/CD ejecuta suite completa en cada PR. Pipeline GitHub Actions: Install → Build → Lint → Test → Coverage Report.

6. Criterios de Entrada y Salida

6.1 Criterios de Entrada

Condiciones que DEBEN cumplirse para iniciar pruebas:

- Código fuente completado y commiteado en rama develop
- Build exitoso sin errores de compilación

- Linter pasando (0 errores críticos)
- Base de datos con schema actualizado y seed data
- Entorno de testing configurado (variables .env.test)
- Dependencias instaladas (npm install completo)
- Documentación de API actualizada (Swagger/OpenAPI)
- Casos de prueba escritos y revisados por QA Lead

6.2 Criterios de Salida

Condiciones para considerar pruebas completadas:

- 100% de casos de prueba ejecutados
- 0 defectos críticos abiertos (bloqueantes)
- <= 2 defectos altos abiertos (con workaround documentado)
- Cobertura de código >= 80% en módulos críticos
- Todos los tests unitarios pasan (100% success rate)
- Todos los tests de integración pasan (100% success rate)
- Flujos end-to-end críticos validados sin errores
- Reporte de pruebas generado y aprobado por QA Lead
- Defectos documentados en GitHub Issues con prioridad asignada

6.3 Criterios de Suspensión

Situaciones que PAUSAN la ejecución de pruebas:

- Entorno de testing inaccesible (servidor caído, BD no responde)
- > 3 defectos críticos encontrados (bloquean flujos principales)
- Cambios mayores en arquitectura o requisitos (requiere replaning)
- Build roto por más de 4 horas
- Falta personal clave (QA Lead enfermo, sin backup)

6.4 Criterios de Reanudación

Condiciones para reanudar pruebas suspendidas:

- Entorno de testing restaurado y estable
- Defectos críticos corregidos y verificados
- Build verde (compilación exitosa)
- Nuevo plan de pruebas aprobado (si hubo cambios de alcance)

- Personal clave disponible o reemplazado

7. Entregables de Pruebas

Entregable	Responsable	Fecha Entrega
Plan de Pruebas (este doc)	QA Lead	Semana 0
Test Cases Postman Collection	QA Team	Semana 1
Unit Tests Suite (Jest)	Developers	Continuo
Test Execution Report	QA Team	Fin de cada sprint
Defect Report (GitHub Issues)	QA + Devs	Continuo
Coverage Report	CI/CD	Cada PR
Test Summary Report	QA Lead	Pre-release
Traceability Matrix	QA Lead	Semana 2
User Acceptance Test Sign-off	Product Owner	Release day

7.1 Formato de Reportes

- Test Execution Report: Excel con columnas [ID, Caso, Estado, Defecto, Evidencia]
- Defect Report: GitHub Issue con template [Título, Pasos, Esperado, Real, Severidad]
- Coverage Report: HTML generado por Jest (jest-html-reporters)
- Test Summary: PDF ejecutivo con métricas [Total, Pass, Fail, Coverage, Defects]

8. Recursos de Pruebas

8.1 Recursos Humanos

Rol	Responsabilidades	Personas	Dedicación
QA Lead	Planificación, revisión, aprobación	1	100%

Backend Dev	Unit tests, corrección de bugs, Ejecución manual, Postman, reportes	1	100%
Tech Lead	Revisión técnica, arquitectura	1	50%
Product Owner	UAT, criterios aceptación	1	50%

8.2 Recursos de Hardware/Software

Hardware:

- Servidores: AWS EC2 t3.medium para testing environment
- Base de datos: PostgreSQL 15 en Supabase (tier gratuito)
- CI/CD: GitHub Actions runners (2 concurrent jobs)

Software:

- Node.js 20 LTS
- Jest 29.x (unit testing)
- Postman Desktop
- PostgreSQL client (psql)
- Git + GitHub
- VS Code + Jest extension

8.3 Herramientas de Prueba

Herramienta	Propósito	Versión
Jest	Unit testing framework	29.7.0
Postman	API testing (manual)	11.x
GitHub Actions	CI/CD pipeline	N/A
jest-html-reporters	Coverage reports	3.x
@nestjs/testing	NestJS test utilities	10.x

9. Matriz de Responsabilidades

Actividad	QA Lead	Tech Lead	PO
Crear Plan de Pruebas	R	A	I
Diseñar Casos de Prueba	A	I	C
Ejecutar Tests Unitarios	I	I	I
Ejecutar Tests Integración	A	I	I
Reportar Defectos	A	I	I
Corregir Defectos	I	A	-
Re-test Correcciones	I	I	-
Generar Reportes	A	I	I
Aprobación UAT	I	I	A
Sign-off Release	R	A	A

Leyenda: R=Responsable, A=Aprueba, C=Consultado, I=Informado

10. Cronograma y Fases de Pruebas

Fase	Actividades	Duración	Inicio	Responsable
Fase 1: Planificación	Crear plan, definir casos, setup entornos	1 semana	Semana 0	QA Lead
Fase 2: Desarrollo Tests	Escribir unit tests, crear colección Postman	1 semanas	Semana 0	Devs + QA
Fase 3: Ejecución Unit	Ejecutar y corregir tests unitarios	1 semana	Semana 1	Developers
Fase 4: Ejecución Integración	Ejecutar Postman, reportar defectos	2 semanas	Semana 3	QA Team
Fase 5: Corrección	Fix bugs, re-testing	1 semana	Semana 3	Devs + QA
Fase 6: Sistema/UAT	Flujos E2E, aceptación usuario	1 semana	Semana 4	QA + PO
Fase 7: Regresión Final	Suite completa pre-release	2 días	Semana 5	CI/CD + QA

Fase 8: Sign-off	Aprobación y documentación final	1 día	Semana 5	PO + Tech Lead

Duración total estimada: 8 semanas

11. Riesgos y Mitigación

Riesgo	Prob	Impacto	Mitigación
Falta de tiempo para testing completo	Alta	Alto	Priorizar casos críticos primero. Automatizar con CI/CD. Involucrar devs en testing.
Entorno de testing inestable	Media	Alto	Usar Docker para entornos reproducibles. Mantener BD de test separada. Monitoring 24/7.
Defectos encontrados tarde (near release)	Media	Alto	Testing continuo desde día 1. Shift-left testing. Code reviews obligatorios.
Baja cobertura de tests unitarios	Media	Medio	Exigir tests en cada PR. CI/CD falla si coverage < 80%. Code review check coverage.
Casos de prueba desactualizados	Media	Medio	Actualizar tests junto con código. Versionamiento de Postman collection. Documentación viva.
Personal QA insuficiente	Baja	Alto	Cross-training de developers. Automatizar máximo posible. Contratar QA freelance si necesario.
Dependencias externas caídas (Supabase)	Baja	Alto	Tests con BD local PostgreSQL. Mocks para servicios externos. Retry logic en CI.

12. Resumen de Casos de Prueba

12.1 Pruebas Unitarias (18 test suites)

Archivo de Prueba	Componente Probado
RegisterUseCase.spec.ts	Registro de usuario con validaciones
LoginUseCase.spec.ts	Login y generación JWT
VerifyEmailUseCase.spec.ts	Verificación de email con token

ForgotPasswordUseCase.spec.ts	Solicitud reset password
ResetPasswordUseCase.spec.ts	Reseteo de contraseña
CreateChildUseCase.spec.ts	Creación de hijo con validaciones
UpdateChildUseCase.spec.ts	Actualización de datos hijo
GetChildrenUseCase.spec.ts	Obtener lista de hijos
DeleteChildUseCase.spec.ts	Eliminación de hijo
AuthController.spec.ts	Controller de autenticación
Email.spec.ts	Value Object Email con regex
Password.spec.ts	Value Object Password con requisitos
UUID.spec.ts	Value Object UUID con validación
User.spec.ts	Entity User con métodos
ChildInfo.spec.ts	Entity ChildInfo con edad 3-12
JwtTokenService.spec.ts	Servicio JWT tokens
BcryptPasswordHasher.spec.ts	Servicio hash contraseñas
SMTPEmailService.spec.ts	Servicio envío emails

Total: ~150-200 test cases individuales dentro de los 18 suites

12.2 Pruebas de Integración (63 casos Postman)

Módulo	Casos	Cobertura
Auth	20	Register (7), Login (3), Refresh (2), Profile (3), Verify (2), Reset (3)
Children	12	Create (4), Update (3), Get (2), Delete (3)
Progress	8	Record attempt (3), Complete lesson (3), Get progress (2)
Content	8	Get levels (2), Get modules (2), Get lessons (2), Get activities (2)
Classes	15	Create (3), Update (2), Delete (1), Join (3), Assignments (4), List (2)

Detalle completo en: POSTMAN_TEST_CASES.md

13. Matriz de Trazabilidad Requisitos-Pruebas

Mapeo entre requisitos funcionales del SRS y casos de prueba que los validan:

Req ID	Requisito Funcional	Casos de Prueba
RF-001	Usuario puede registrarse con email y contraseña	RegisterUseCase.spec.ts (Unit), POST /auth/register casos 1.1.1-1.1.7 (Integration)
RF-002	Usuario puede iniciar sesión y obtener JWT	LoginUseCase.spec.ts (Unit), POST /auth/login casos 1.2.1-1.2.3 (Integration)
RF-003	Sistema valida formato de email con regex	Email.spec.ts (Unit), Casos 1.1.4 email inválido (Integration)
RF-004	Contraseña requiere min 8 chars, mayúscula, número, especial	Password.spec.ts (Unit), Casos 1.1.5 password débil (Integration)
RF-005	Padre puede registrar hasta 3 hijos (plan FREE)	CreateChildUseCase.spec.ts (Unit), POST /children casos 2.1.1-2.1.4 (Integration)
RF-006	Edad de hijo debe estar entre 3-12 años	ChildInfo.spec.ts (Unit), Caso 2.1.2 edad inválida (Integration)
RF-007	Sistema registra intentos de actividad con accuracy	POST /progress/attempt casos 3.1.1-3.1.3 (Integration)
RF-008	Lección se completa con accuracy >= 80%	POST /progress/complete-lesson caso 3.2.1 (Integration)
RF-009	Nivel se desbloquea al completar 80% lecciones	Validado en flujo E2E de sistema
RF-010	Docente puede crear clase con código único 6 chars	POST /classes casos 5.1.1-5.1.3 (Integration)
RF-011	Padre puede unirse a clase con código	POST /classes/join casos 5.6.1-5.6.3 (Integration)
RF-012	Contenido organizado en: Niveles → Módulos → Lecciones	GET /content/* casos 4.2.1, 4.4.1, 4.5.1 (Integration)

Cobertura: 12 de 12 requisitos funcionales críticos cubiertos (100%)

14. Aprobaciones

Este Plan de Pruebas debe ser revisado y aprobado por los siguientes roles antes de iniciar la ejecución:

Rol	Nombre	Firma	Fecha
QA Lead	Fernando Tipán	_____	__/__/____
Tech Lead	Mateo Sosa	_____	__/__/____
Product Owner	Marcos Escobar	_____	__/__/____

Historial de Aprobaciones

Versión	Fecha	Aprobador	Comentarios
1.0	19/1/2026	Marcos Escobar	Versión inicial para revisión