Project Spec — Plataforma de Simulacros tipo PISA (Android + Firebase + Backoffice + IA)

> Documento maestro (single source of truth) para que una IA agéntica pueda **planificar y generar código** sin ambigüedades. Alcance: **MVP funcional completo** + bases para escalar a v2.

---

0) Resumen ejecutivo

Se desarrollará una plataforma educativa compuesta por:

- **App Android (Jetpack Compose)**: simulacros tipo PISA (lecturas largas + preguntas), gamificación con EduCoins, tienda cosmética, perfil/estadísticas, logros, ajustes de accesibilidad (tamaño de fuente), notificaciones locales.
- **Modo Offline-first real**: el alumno descarga el **"Pack de la Semana"** y lo guarda en **Room**. El examen se ejecuta **100% desde Room** incluso si pasan semanas sin internet.
- **Sincronización híbrida Room ↔ Firestore**: la app escribe/lee local; un **WorkManager** sincroniza con **Cloud Firestore** cuando hay internet.
- **Backoffice Web Docente (obligatorio)**: carga de textos, administración de usuarios/aulas y métricas; generación/gestión de contenido para poblar la base.
- **IA (Groq) vía Cloud Functions**:
  - Genera **preguntas** (desde backoffice).
  - Genera **explicaciones/feedback** (pre-cargado y reutilizable por `question_id` → "dataset global").

---

1) Decisiones cerradas (no debatibles en este documento)

Autenticación y acceso

- **Login obligatorio (Hard Gate)**: no existe modo invitado.
- Auth: **Firebase Auth** con **Google Sign-In**.
- No se implementa modo kiosco ni multi-perfil local.

Offline y datos

- Offline del contenido de examen: **Room + paquetes descargables** (Pack de la Semana).
- Datos híbridos: **Room como lectura/escritura principal** + **Sync a Cloud Firestore** con WorkManager.

Resolución de conflictos de sync (por entidad)

- **Puntajes/Exámenes**: *merge/unión* (nunca borrar, siempre sumar al historial).
- **Perfil/Avatar**: *última escritura gana*.

Core Loop del examen

- Modo principal: **Simulacro PISA por bloques**
  - Un texto largo asociado a **3–5 preguntas**.
  - Un examen = **2 textos → 10 preguntas**.
- Timer: **20 minutos por examen (global)**.
- Fin de tiempo: **auto-submit**, se califica lo respondido, no se pierde lo marcado.

Gamificación

- EduCoins por respuestas correctas.
- Bonus por:
  - **racha** (3 días seguidos entrando),
  - **velocidad** (responder en < 1 minuto).
- Tienda: cosmética básica (avatar/skins/frames/íconos).

Seguridad anti-trampa (durante examen)

- **Screenshots bloqueadas solo durante el examen** (FLAG_SECURE).
- **Bloqueo de botones 5s** al cargar cada pregunta.
- Salir/cambiar de app:
  - 1ª vez: advertencia,
  - 2ª vez: se anula/cierra el examen.

Notificaciones

- Solo **locales/programadas** (recordatorios de estudio). Push complejas fuera del MVP.

Ranking

- Solo **Ranking de Aula/Colegio** (no ranking nacional en MVP).
- Ranking como **documento agregado** (leaderboard), calculado/actualizado en backend.

IA

- Proveedor: **Groq**.
- Se guarda feedback por `question_id` (misma explicación para todos).

Accesibilidad

- Ajuste de tamaño de fuente: **sí (vital)**.
- TTS: **no en MVP**.

Backoffice

- **Obligatorio**: sin backoffice la DB quedaría vacía.
- Funciones: subir textos, métricas por aula, administrar usuarios/aulas.
- Cloud Functions **sí**: para IA (ocultar API Key) y ranking pesado.

---

2) Objetivos del MVP

1. Permitir que un alumno autenticado descargue el **Pack de la Semana**.
2. Ejecutar un simulacro completo offline desde Room.
3. Guardar resultados localmente y sincronizarlos a Firestore cuando haya internet.
4. Mostrar perfil básico (historial), logros, tienda cosmética y ajustes de accesibilidad.
5. Prevenir trampas básicas durante el examen.
6. Permitir a un docente/admin cargar textos y gestionar contenido/aulas en backoffice.
7. Generar contenido asistido por IA (preguntas + explicaciones pre-cargadas) desde backoffice.

---

3) Actores y roles

- **Alumno**
  - Realiza simulacros, gana monedas, compra cosméticos, consulta historial/logros.
- **Docente/Admin (Backoffice)**
  - Carga textos, solicita generación de preguntas/explicaciones, publica packs.
  - Administra aulas/usuarios y consulta métricas.
- **Sistema (Cloud Functions)**
  - Orquesta IA, valida/calcula puntajes, actualiza leaderboards.

---

4) Requisitos funcionales (alto nivel)

App Android

- Auth obligatorio (Google Sign-In).
- Descargar y gestionar “Pack de la Semana”.
- Tomar simulacro PISA (texto + preguntas) con timer 20 min.
- Guardar progreso y resultados (Room).
- Sync diferido con Firestore (WorkManager).
- Perfil: historial simple + avatar + monedas.
- Logros.
- Tienda cosmética.
- Ajustes: tamaño de fuente.
- Notificaciones locales programables.

- Ranking por aula.

Backoffice Web

- CRUD Textos.
- Generación IA:
    - Generar preguntas por texto.
    - Generar explicaciones por pregunta.
- Publicar Pack de la Semana.
- Admin aulas/usuarios.
- Métricas por aula.

Backend (Firebase)

- Firestore como base central.
- Cloud Functions:
    - Proxy Groq + persistencia de resultados.
    - Cálculo/actualización de leaderboard.
    - (Opcional MVP+) Validación de puntaje server-side.

---

5) Arquitectura recomendada

5.1 Android: Clean Architecture + MVVM

Capas:

- **Presentation** (Compose + ViewModel)
- **Domain** (UseCases + interfaces)
- **Data** (Repositorios + Room + Firestore + mappers)
- **Core** (utilidades comunes: Result, Dispatchers, logger, etc.)

Patrón:

- Repositorio único por feature (perfil, pack, examen, tienda, ranking).
- La UI **nunca** accede a Room/Firestore directo.

5.2 Sync: Offline-first

- La app opera sobre Room.
- Cambios se registran con `sync_state = PENDING`.
- Un `WorkManager`:
    - detecta internet,
    - sube deltas,
    - resuelve conflictos por regla,
    - marca `sync_state = SYNCED`.

---

6) Estructura del repositorio (monorepo)

/root /android /app /core /data /domain /feature-auth /feature-pack /feature-exam /feature-profile /feature-store /feature-ranking /functions (Firebase Cloud Functions - TypeScript) /web-admin (Backoffice - Next.js o React) /docs PROJECT_SPEC.md (este archivo)

yaml Copiar código

---

7) Modelo de datos (fuente de verdad)

> Regla: **Contenido de examen** vive en Firestore (authoring) → se descarga a Room como pack. Resultado/usuario: Room primero, luego sync a Firestore.

---

8) Firestore — Colecciones y documentos

8.1 Contenido (global)

/content_texts/{textId} /content_questions/{questionId} /packs/{packId}

pgsql Copiar código

content_texts/{textId}

```
{
  "textId": "txt_2025_w01_001",
  "title": "La energía solar en ciudades",
  "body": "Texto largo...",
  "subject": "LECTURA|MATEMATICA|CIENCIAS",
  "gradeBand": "PISA",
  "createdAt": "timestamp",
  "updatedAt": "timestamp",
  "status": "DRAFT|APPROVED|PUBLISHED"
```

}

content_questions/{questionId}

json

Copiar código

```
{
  "questionId": "q_2025_w01_0001",
  "textId": "txt_2025_w01_001",
  "prompt": "¿Cuál es la idea principal...?",
  "options": [
    { "optionId": "A", "text": "..." },
    { "optionId": "B", "text": "..." },
    { "optionId": "C", "text": "..." },
    { "optionId": "D", "text": "..." }
  ],
  "correctOptionId": "B",
  "difficulty": 1,
  "tags": ["inferencia", "comprension"],
  "explanation": {
    "status": "NONE|GENERATED|APPROVED",
    "text": "B es correcta porque..."
  },
  "createdAt": "timestamp",
  "updatedAt": "timestamp",
  "status": "DRAFT|APPROVED|PUBLISHED"
```

}

packs/{packId} (Pack de la Semana)

json

Copiar código

```
{
  "packId": "pack_2025_w01",
  "weekLabel": "2025-W01",
  "subjects": ["LECTURA", "MATEMATICA", "CIENCIAS"],
  "textIds": ["txt_2025_w01_001", "txt_2025_w01_002", "..."],
  "questionIds": ["q_2025_w01_0001", "q_2025_w01_0002", "..."],
  "publishedAt": "timestamp",
  "status": "PUBLISHED"
}
```

8.2 Usuarios y aulas

bash

Copiar código

```
/users/{uid}
/schools/{schoolId}
/schools/{schoolId}/classrooms/{classroomId}
/schools/{schoolId}/classrooms/{classroomId}/leaderboard/{doc}
```

users/{uid}

json

Copiar código

```
{
```

```
  "uid": "firebase_uid",
  "displayName": "Juan",
  "photoUrl": "https://...",
  "schoolId": "sch_001",
  "classroomId": "cls_003",
  "coins": 120,
  "selectedCosmeticId": "cos_01",
  "createdAt": "timestamp",
  "updatedAt": "timestamp"
}
```

users/{uid}/examAttempts/{attemptId}

json

Copiar código

```
{
  "attemptId": "att_2025_12_08_0001",
  "packId": "pack_2025_w01",
  "startedAt": "timestamp",
  "finishedAt": "timestamp",
  "durationMs": 1134000,
  "status": "COMPLETED|AUTO_SUBMIT|CANCELLED_CHEAT",
  "scoreRaw": 7,
  "scoreValidated": 7,
  "subjectBreakdown": { "LECTURA": 3, "MATEMATICA": 2, "CIENCIAS": 2 },
  "answers": [
```

```
    { "questionId": "q_...", "selectedOptionId": "B", "isCorrect": true, "timeSpentMs": 42000 }
  ],
  "syncMeta": { "origin": "OFFLINE|ONLINE", "appVersion": "1.0.0" }
}
```

Leaderboard (agregado por aula)

/schools/{schoolId}/classrooms/{classroomId}/leaderboard/current

json

Copiar código

```
{
  "classroomId": "cls_003",
  "updatedAt": "timestamp",
  "top": [
    { "uid": "u1", "displayName": "Ana", "score": 120 },
    { "uid": "u2", "displayName": "Luis", "score": 110 }
  ]
}
```

9) Room — Tablas y esquema exacto

Nota: Room es la base operativa. Firestore se sincroniza de forma eventual.

9.1 Entidades principales

pack_entity

packId: String (PK)

weekLabel: String

status: String (DOWNLOADED/ACTIVE/ARCHIVED)

publishedAt: Long

downloadedAt: Long

text_entity

textId: String (PK)

packId: String (FK)

title: String

body: String

subject: String (LECTURA/MATEMATICA/CIENCIAS)

question_entity

questionId: String (PK)

packId: String (FK)

textId: String (FK)

prompt: String

correctOptionId: String

difficulty: Int

explanationText: String? (pre-cargado)

explanationStatus: String (NONE/GENERATED/APPROVED)

option_entity

questionId: String (PK part)

optionId: String (PK part)

text: String

9.2 Usuario / progreso / economía

user_profile_entity (single row por uid)

uid: String (PK)

displayName: String

photoUrl: String?

schoolId: String

classroomId: String

coins: Int

selectedCosmeticId: String

updatedAtLocal: Long

syncState: String (PENDING/SYNCED)

inventory_entity

uid: String (PK part)

cosmeticId: String (PK part)

purchasedAt: Long

achievement_entity

uid: String (PK part)

achievementId: String (PK part)

unlockedAt: Long

daily_streak_entity

uid: String (PK)

currentStreak: Int

lastLoginDate: String (YYYY-MM-DD)

updatedAtLocal: Long

syncState: String

9.3 Exámenes

exam_attempt_entity

attemptId: String (PK)

uid: String

packId: String

startedAtLocal: Long

finishedAtLocal: Long?

durationMs: Long

status: String (IN_PROGRESS/COMPLETED/AUTO_SUBMIT/CANCELLED_CHEAT)

scoreRaw: Int

scoreValidated: Int?

origin: String (OFFLINE/ONLINE)

syncState: String (PENDING/SYNCED/FAILED)

exam_answer_entity

attemptId: String (PK part)

questionId: String (PK part)

selectedOptionId: String

isCorrect: Boolean

timeSpentMs: Long

10) Reglas de sincronización (Room → Firestore)

10.1 WorkManager

Worker periódico (por ejemplo cada 6h) + worker one-shot al detectar internet.

Constraints: NetworkType.CONNECTED.

Reintentos exponenciales.

10.2 Estrategia por entidad

Exam attempts:

Subir intentos syncState=PENDING.

En backend (Function) recalcular scoreValidated si aplica.

Merge: nunca borrar intentos.

Perfil (avatar/selectedCosmeticId):

Última escritura gana usando updatedAtLocal como tiebreaker.

Coins:

En MVP, coins se sincronizan como valor final calculado localmente + validación opcional server-side.

Recomendado: registrar “ledger” de transacciones (v2).

11) Flujo del “Pack de la Semana”

Docente/Admin publica un packs/{packId} con listas de textIds y questionIds.

App Android descarga:

pack doc,

textos referenciados,

preguntas + opciones + explicación (si existe).

Inserta todo en Room en una transacción:

pack_entity

text_entity

question_entity

option_entity

Marca pack_entity.status = ACTIVE.

Regla: Durante el examen NO se lee Firestore. Solo Room.

12) Seguridad anti-trampa (especificación exacta)

12.1 Screenshots

En pantallas del examen: activar FLAG_SECURE.

En perfil/tienda: desactivado (permitir marketing orgánico).

12.2 Bloqueo 5s

Al cargar una pregunta:

deshabilitar opciones por 5s,

mostrar indicador “Analizando...” o “Lee con atención”.

12.3 Salir de la app

Detectar onStop/onPause del flujo de examen.

1ª vez:

mostrar modal: “Si sales una vez más, se anula el examen”.

guardar leaveCount = 1.

2ª vez:

cerrar examen y marcar intento como CANCELLED_CHEAT.

opcional: guardar respuestas ya marcadas y subir con estado cancelado.

12.4 Timer

Usar SystemClock.elapsedRealtime() como fuente.

Límite: 20 * 60 * 1000 ms.

Auto-submit cuando expire.

13) IA (Groq) — Funciones, prompts y caché

13.1 Principio clave

Explicaciones y preguntas se reutilizan globalmente.

Explicación indexada por question_id.

13.2 Cloud Functions (TypeScript)

Funciones mínimas:

generateQuestionsForText(textId)

Entrada: textId

Acción: llama Groq, genera N preguntas (3–5), guarda en /content_questions.

Estado: DRAFT o APPROVED según flujo.

generateExplanationForQuestion(questionId)

Entrada: questionId

Acción: llama Groq, genera explicación de la respuesta correcta, guarda en explanation.text y explanation.status=GENERATED.

recomputeAttemptScore(uid, attemptId)

Entrada: intento subido por sync.

Acción: valida respuestas con correctOptionId server-side.

Guarda scoreValidated y actualiza leaderboard.

13.3 Prompts (plantillas)

Prompt para preguntas (ejemplo)

Sistema: “Eres un generador de preguntas tipo PISA…”

Usuario: incluir texto, materia, cantidad, formato JSON estricto.

Prompt para explicación (ejemplo)

Sistema: “Explica por qué la opción correcta es correcta…”

Usuario: incluir prompt, opciones, correctOptionId, y un estilo breve.

14) Backoffice Web (MVP obligatorio)

14.1 Funcionalidades

Login admin/docente (Firebase Auth).

CRUD de textos (content_texts).

Gestión de preguntas por texto:

crear/editar manual,

generar con IA (Cloud Function),

aprobar/publicar.

Publicar Pack de la Semana (packs).

Admin de escuelas/aulas/usuarios (asignación schoolId, classroomId).

Métricas:

intentos por aula,

promedio por materia,

evolución semanal.

14.2 Stack sugerido

Next.js + Firebase SDK (Auth + Firestore).

UI: Material UI o similar.

Llamadas a Functions por HTTPS Callable.

15) Android — Estructura de features y responsabilidades

Features

feature-auth: login Google, sesión, logout.

feature-pack: listar pack actual, descargar, estado local.

feature-exam: flow del simulacro, timer, anti-trampa.

feature-profile: historial, avatar, stats simples.

feature-store: catálogo cosmético y compras.

feature-ranking: ranking por aula.

Repositorios (firmas sugeridas)

kotlin

Copiar código

```
interface AuthRepository {
  suspend fun signInWithGoogle(idToken: String): Result<Unit>
```

```
  suspend fun signOut(): Result<Unit>

  fun currentUserId(): String?

}

interface PackRepository {

  suspend fun fetchCurrentPackMeta(): Result<PackMeta>

  suspend fun downloadPack(packId: String): Result<Unit>

  fun observeActivePack(): Flow<PackEntity?>

}

interface ExamRepository {

  suspend fun startAttempt(packId: String): Result<String /*attemptId*/>

  suspend fun submitAnswer(attemptId: String, questionId: String, optionId: String,
timeSpentMs: Long): Result<Unit>

  suspend fun finishAttempt(attemptId: String, status: AttemptStatus): Result<Unit>

  fun observeAttempt(attemptId: String): Flow<ExamAttemptEntity>

}

interface ProfileRepository {

  fun observeProfile(uid: String): Flow<UserProfileEntity?>

  suspend fun updateSelectedCosmetic(cosmeticId: String): Result<Unit>

  suspend fun addCoins(delta: Int, reason: String): Result<Unit>

}

interface SyncRepository {

  suspend fun enqueueSyncNow(): Result<Unit>

}
```

16) Dependencias Android (build.gradle) — lista mínima

Nota práctica: se recomienda Version Catalog (libs.versions.toml) para mantener versiones consistentes.

Aquí se listan dependencias por coordenadas; la IA agéntica puede fijar versiones estables.

Jetpack Compose (UI)

Lifecycle + ViewModel

Navigation Compose

Room (offline DB)

WorkManager (sync)

Hilt (DI)

Firebase BOM:

Auth

Firestore

Google Sign-In

Kotlin Coroutines

Kotlin Serialization o Moshi (para parsear payloads del pack)

Ejemplo (gradle Kotlin DSL, sin versiones explícitas):

kotlin

Copiar código

```
dependencies {
  // Compose
  implementation(platform("androidx.compose:compose-bom:VERSION_BOM"))
  implementation("androidx.compose.ui:ui")
  implementation("androidx.compose.material3:material3")
  implementation("androidx.navigation:navigation-compose:VERSION_NAV")
  // Lifecycle / ViewModel
```

```
  implementation("androidx.lifecycle:lifecycle-runtime-ktx:VERSION_LIFECYCLE")
  implementation("androidx.lifecycle:lifecycle-viewmodel-compose:VERSION_LIFECYCLE")
  // Room
  implementation("androidx.room:room-runtime:VERSION_ROOM")
  kapt("androidx.room:room-compiler:VERSION_ROOM")
  implementation("androidx.room:room-ktx:VERSION_ROOM")
  // WorkManager
  implementation("androidx.work:work-runtime-ktx:VERSION_WORK")
  // DI
  implementation("com.google.dagger:hilt-android:VERSION_HILT")
  kapt("com.google.dagger:hilt-compiler:VERSION_HILT")
  // Firebase

implementation(platform("com.google.firebase:firebase-bom:VERSION_FIREBASE_BOM"))
  implementation("com.google.firebase:firebase-auth-ktx")
  implementation("com.google.firebase:firebase-firestore-ktx")
  // Google Sign-In
  implementation("com.google.android.gms:play-services-auth:VERSION_GMS_AUTH")
  // Coroutines

implementation("org.jetbrains.kotlinx:kotlinx-coroutines-android:VERSION_COROUTINES")
}
```

17) Criterios de aceptación (MVP)

Offline Pack

Sin internet, el alumno puede iniciar y terminar un examen si el pack ya fue descargado.

El examen usa exclusivamente datos locales.

Los resultados quedan en Room con syncState=PENDING.

Sync

Al recuperar internet, WorkManager sube intentos pendientes.

Firestore refleja el historial completo (merge).

Perfil se sincroniza con “última escritura gana”.

Seguridad

Capturas bloqueadas durante examen.

Botones bloqueados 5s por pregunta.

Salir 2 veces anula intento.

Backoffice

Docente puede crear texto, generar preguntas con IA, aprobar y publicar pack.

La app descarga el pack publicado.

Métricas por aula visibles.

18) Roadmap (v2 sugerida, fuera del MVP)

Push Notifications (FCM).

TTS.

Ranking nacional.

Ledger de coins server-side + antifraude avanzado.

Export “pack JSON” optimizado (Cloud Storage) para descargas masivas.

19) Convenciones (nombres y estándares)

IDs:

pack_YYYY_wWW

txt_...

q_...

att_...

Estados:

syncState: PENDING/SYNCED/FAILED

question.status: DRAFT/APPROVED/PUBLISHED

Materias:

LECTURA, MATEMATICA, CIENCIAS

20) Checklist de implementación (orden recomendado)

Firebase Auth + login gate.

Room schema + DAOs.

Descarga de pack desde Firestore → inserción en Room.

Flow de examen (timer + anti-trampa + auto-submit).

Persistencia de intentos y respuestas en Room.

WorkManager sync de intentos a Firestore.

Perfil + coins + tienda.

Logros + racha.

Ranking aula (Cloud Function + documento agregado).

Backoffice (CRUD textos, publicar pack, métricas).

Cloud Functions IA (Groq) + cache global por question_id.

pgsql

Copiar código