## **¿Qué es? y como utilizar appMovi (Proyecto Ingeniería)**

Alisson Bonilla

Natalia

Brayanne Roberto Moreno Ruiz

Universidad Antonio Nariño

Facultad de Ingeniería de Sistemas y Computación

Nombre Profesor, Materia  
Bogotá D.C.

2025

# **Tabla de contenido**

Contenido

[**¿Qué es? y como utilizar appMovi (Proyecto Ingeniería)** 1](#_Toc210590421)

[**Tabla de contenido** 2](#_Toc210590422)

[1. RDe 4](#_Toc210590423)

[2. Introducción 5](#_Toc210590424)

[3. Descripción general del proyecto 6](#_Toc210590425)

[4. Objetivos del proyecto 7](#_Toc210590426)

[4.1 Objetivos General 7](#_Toc210590427)

[4.2 Objetivos Especifico 7](#_Toc210590428)

[5. Análisis y requisitos 8](#_Toc210590429)

[5.1 Requerimientos Funcionales 8](#_Toc210590430)

[5.2 Requerimientos de Interfaz 8](#_Toc210590431)

[5.3 Requisitos de Rendimiento 9](#_Toc210590432)

[5.4 Requisitos No Funcionales 9](#_Toc210590433)

[6. Historias de Usuario y Criterios de Aceptación 11](#_Toc210590434)

[**👤** Rol: Jugador 11](#_Toc210590435)

[**🛠️** Rol: Administrador 14](#_Toc210590436)

[**🤖** Rol: Inteligencia Artificial 15](#_Toc210590437)

[7. Casos de Uso del Sistema 16](#_Toc210590438)

[7.1 Actores del Sistema 16](#_Toc210590439)

[7.2 Descripción de Casos de Uso 16](#_Toc210590440)

[8. Arquitectura de Software 17](#_Toc210590441)

[8.1 Principios SOLID aplicados 17](#_Toc210590442)

[8.2 Patrón arquitectónico: MVVM (y mención de Hexagonal como alternativa) 17](#_Toc210590443)

[8.3 Microservicios utilizados (solo si se integran APIs externas) 17](#_Toc210590444)

[8.4 Diagramas C4 (Nivel de contexto, contenedores y componentes) 17](#_Toc210590445)

[9. Modelado Estructural 18](#_Toc210590446)

[9.1 Diagrama de Clases (UML) 18](#_Toc210590447)

[**9.2 Diagrama de Objetos** (opcional para este alcance) 18](#_Toc210590448)

[9.3 Diagrama de Componentes 18](#_Toc210590449)

[**9.4 Diagrama de Despliegue** (para mostrar cómo se ejecuta en Android con Firebase) 18](#_Toc210590450)

[**9.5 Diagrama de Paquetes** (opcional, según la arquitectura MVVM) 18](#_Toc210590451)

[10. Modelado de Comportamiento 19](#_Toc210590452)

[10.1 Diagramas de Casos de Uso (Jugador, Administrador, Sistema) 19](#_Toc210590453)

[10.2 Diagrama de Actividad 19](#_Toc210590454)

[10.3 Diagrama de Estado *(para flujo del jugador o partida)* 19](#_Toc210590455)

[10.4 Diagrama de Secuencia 19](#_Toc210590456)

[11. Diagramas de Arquitectura 20](#_Toc210590457)

[11.1 Diagrama de Arquitectura de Software 20](#_Toc210590458)

[11.2 Diagrama de Arquitectura del Sistema 20](#_Toc210590459)

[11.3 Diagrama de Arquitectura de Aplicaciones 20](#_Toc210590460)

[11.4 Diagrama de Arquitectura de Integración (IA - Firebase) 20](#_Toc210590461)

[11.5 Diagrama de Arquitectura de Implementación (Kotlin - XML) 20](#_Toc210590462)

[12. Escenarios Operativos 21](#_Toc210590463)

[13. Cronograma o Calendario Actualizado 22](#_Toc210590464)

[14. Conclusiones 23](#_Toc210590465)

[15. Bibliografía 24](#_Toc210590466)

# RDe

El propósito de este documento es definir y describir los requisitos del proyecto, explicar la funcionalidad del sistema y sus limitaciones. Incluyendo la fecha límite para la entrega del documento, que vence el 12/12/1111.

# Introducción

El presente documento describe el desarrollo del proyecto RDE, una aplicación móvil educativa en forma de videojuego 3D, creada con el propósito de enseñar a los niños cómo actuar correctamente frente a un terremoto. El proyecto está desarrollado en Kotlin, sin el uso de Jetpack Compose, e integra servicios de inteligencia artificial y Firebase para ofrecer una experiencia interactiva, dinámica y con persistencia de datos.  
A lo largo del documento se presentan los objetivos, requisitos, arquitectura, modelado y aspectos técnicos que definen la estructura y funcionamiento de la aplicación, con el fin de garantizar un diseño eficiente, mantenible y alineado con buenas prácticas de ingeniería de software.

# Descripción general del proyecto

RDE es una aplicación móvil educativa e interactiva desarrollada en Kotlin, cuyo propósito es enseñar a los niños cómo reaccionar adecuadamente frente a un terremoto. A través de un juego 3D, los usuarios participan en una historia inicial común que evoluciona según sus respuestas en trivias educativas y decisiones tomadas durante el juego.

La aplicación integrará inteligencia artificial (IA) para generar dinámicamente imágenes y posibles escenarios, personalizando la experiencia del jugador y haciendo el aprendizaje más inmersivo. Además, contará con funcionalidades de registro, carga, creación y reanudación de partidas, asegurando la persistencia de datos para simular el comportamiento de un sistema real.

# Objetivos del proyecto

El proyecto RDE tiene como objetivo principal desarrollar un videojuego educativo interactivo que combine entretenimiento y aprendizaje, incentivando el desarrollo cognitivo y la toma de decisiones del jugador mediante mecánicas dinámicas y visualmente atractivas. A través de este desarrollo, se busca aplicar los principios de la ingeniería de software, abarcando desde el análisis y diseño hasta la implementación y pruebas, garantizando una experiencia de usuario sólida y un producto funcional acorde con los estándares académicos y profesionales.

### 4.1 Objetivos General

Desarrollar una aplicación móvil interactiva con integración de inteligencia artificial que promueva el aprendizaje infantil sobre cómo actuar ante un terremoto mediante un entorno 3D educativo y dinámico.

### Objetivos Especifico

* + 1. Implementar un sistema de registro e inicio de sesión básico para identificar a cada jugador.
    2. Diseñar una interfaz 3D interactiva que combine elementos educativos y de entretenimiento.
    3. Integrar trivias y escenarios variables que evalúen el conocimiento adquirido por el jugador.
    4. Incorporar una funcionalidad de IA para la generación de imágenes o escenarios personalizados según el progreso del usuario.
    5. Garantizar la persistencia de datos para guardar y reanudar partidas.
    6. Facilitar la navegación por categorías de entrenamiento, comenzando con el modo terremoto.

# 5. Análisis y requisitos

El proyecto RDE tiene como objetivo principal desarrollar un videojuego educativo interactivo que combine entretenimiento y aprendizaje, incentivando el desarrollo cognitivo y la toma de decisiones del jugador mediante mecánicas dinámicas y visualmente atractivas. A través de este desarrollo, se busca aplicar los principios de la ingeniería de software, abarcando desde el análisis y diseño hasta la implementación y pruebas, garantizando una experiencia de usuario sólida y un producto funcional acorde con los estándares académicos y profesionales.

### 5.1 Requerimientos Funcionales

1. El sistema debe permitir al usuario ingresar su nombre e iniciar sesión.

2. El usuario podrá seleccionar una categoría de entrenamiento (por defecto, “modo terremoto”).

3. El sistema debe permitir crear, cargar y reanudar partidas guardadas.

4. La aplicación debe presentar trivias y actividades educativas durante el desarrollo del juego.

5. La IA debe generar imágenes o escenarios 3D basados en las decisiones o resultados de las trivias del jugador.

6. El sistema debe guardar el progreso y permitir consultar el historial de partidas.

7. El usuario podrá reanudar o salir del juego en cualquier momento.

### 5.2 Requerimientos de Interfaz

1. La interfaz debe ser intuitiva, colorida y adaptada al público infantil.
2. El diseño debe incluir botones y elementos visuales grandes y fácilmente reconocibles.
3. Las pantallas principales deben incluir:
   1. Pantalla de inicio con acceso a registro o carga de partida.
   2. Menú de selección de categoría.
   3. Escena 3D del juego.
   4. Pantalla de trivias.
   5. Pantalla de resultados o resumen de aprendizaje.
4. Los elementos visuales (fondos, personajes, escenarios) deben ser coherentes con el entorno de un terremoto y sus enseñanzas.
5. La interfaz debe integrarse adecuadamente con las funciones de IA para mostrar imágenes o escenas generadas dinámicamente.

### 5.3 Requisitos de Rendimiento

1. La aplicación debe ser capaz de cargar el entorno 3D y los recursos multimedia (modelos, texturas y audio) en un tiempo máximo de 3 segundos en dispositivos de gama media.
2. El sistema debe mantener una tasa mínima de 25 FPS durante la ejecución del juego.
3. Las peticiones a los servicios de inteligencia artificial y Firebase deben procesarse en menos de 2 segundos en condiciones normales de red.
4. El uso de memoria RAM no debe exceder el 70 % de la capacidad disponible del dispositivo mientras el juego está en ejecución.

### 5.4 Requisitos No Funcionales

Los requisitos no funcionales definen las características de calidad y comportamiento general del sistema, estableciendo cómo debe operar la aplicación más allá de sus funciones principales. En el caso del proyecto RDE, estos requisitos garantizan la seguridad, rendimiento, mantenibilidad y eficiencia del videojuego educativo, asegurando una experiencia estable y confiable para el usuario final.

#### 5.4.1 Seguridad

1. El acceso a los datos de usuario (nombre, progreso, partidas) debe gestionarse mediante autenticación segura de Firebase.
2. La aplicación no debe almacenar información sensible en texto plano.
3. Se implementará validación de entrada en formularios para evitar inyección o manipulación de datos.

#### 5.4.2 Fiabilidad

1. La aplicación debe mantener la consistencia del progreso del jugador, guardando automáticamente antes de salir o ante cierres inesperados.
2. Los errores de conexión deben manejarse mediante mensajes de retroalimentación y opción de reintento.

#### 5.4.3 Mantenibilidad

1. El código debe organizarse siguiendo la arquitectura MVVM, separando responsabilidades entre la interfaz, la lógica de negocio y los datos.
2. Se deben emplear nombres descriptivos y comentarios claros para facilitar futuras actualizaciones o correcciones.

#### 5.4.4 Portabilidad

1. La aplicación debe poder ejecutarse en dispositivos Android 8.0 (API 26) o superior.
2. No se requiere compatibilidad con otros sistemas operativos.

#### 5.4.5 Extensibilidad

1. La arquitectura del sistema debe permitir agregar nuevas categorías de desastres naturales (por ejemplo, incendios o inundaciones) sin alterar el núcleo del juego.

#### 5.4.6 Reutilización

1. Los módulos de gestión de usuarios, almacenamiento y comunicación con IA deben poder reutilizarse en futuras versiones o proyectos similares.

#### Utilización de recursos

1. El sistema debe optimizar el uso de CPU y GPU durante la ejecución de gráficos 3D.

2. Las conexiones con IA deben limitarse a peticiones esenciales, evitando consumo innecesario de red o energía.

# 6. Historias de Usuario y Criterios de Aceptación

**👤** Rol: Jugador

**HU1 – Registro o ingreso al juego**

**Como** jugador,  
**quiero** ingresar al juego utilizando mi nombre o alias,  
**para** que el sistema guarde mi progreso y personalice mi experiencia.

**Criterios de aceptación:**

* El jugador debe poder ingresar un nombre o alias antes de iniciar el juego.
* El sistema debe validar que el campo no esté vacío.
* Al confirmar, se debe crear o cargar una partida vinculada al jugador.
* El sistema debe mostrar un mensaje de bienvenida con el nombre del jugador.
* Fin del criterio de aceptación.

**HU2 – Selección de categoría de entrenamiento**

**Como** jugador,  
**quiero** seleccionar una categoría (por ejemplo, modo terremoto),  
**para** iniciar una partida relacionada con ese escenario educativo.

**Criterios de aceptación:**

* El sistema debe mostrar las categorías disponibles.
* Al seleccionar “Modo Terremoto”, debe cargarse la historia y los retos asociados.
* El sistema debe permitir agregar futuras categorías sin alterar el flujo actual.
* Fin del criterio de aceptación.

**HU3 – Iniciar nueva partida**

**Como** jugador,  
**quiero** crear una nueva partida,  
**para** comenzar desde el inicio del juego y registrar mi progreso.

**Criterios de aceptación:**

* El sistema debe mostrar un botón o menú para crear nueva partida.
* Al crearla, se deben inicializar las variables de progreso del jugador.
* El sistema debe confirmar la creación con un mensaje visual.
* Fin del criterio de aceptación.

**HU4 – Reanudar partida guardada**

**Como** jugador,  
**quiero** reanudar mi partida guardada,  
**para** continuar el progreso desde el punto donde la dejé.

**Criterios de aceptación:**

* El jugador debe poder consultar una lista de partidas guardadas.
* El sistema debe cargar correctamente el estado de la partida (nivel, respuestas, etc.).
* Si no hay partidas guardadas, debe mostrar un mensaje informativo.
* Fin del criterio de aceptación.

**HU5 – Jugar y resolver trivias**

**Como** jugador,  
**quiero** responder trivias o preguntas interactivas,  
**para** aprender las acciones correctas en situaciones de emergencia.

**Criterios de aceptación:**

* El sistema debe presentar preguntas con opciones de respuesta.
* Al responder, debe mostrar retroalimentación inmediata (correcta/incorrecta).
* Las respuestas correctas deben influir en la evolución de la historia.
* Fin del criterio de aceptación.

**HU6 – Recibir retroalimentación dinámica**

**Como** jugador,  
**quiero** recibir mensajes o imágenes generadas por IA,  
**para** visualizar las consecuencias o aprendizajes de mis decisiones.

**Criterios de aceptación:**

* El sistema debe realizar una petición al servicio de IA (por ejemplo, Play.ht o similar).
* La IA debe generar imágenes o contenido coherente con la decisión del jugador.
* Si hay error en la generación, el sistema debe mostrar un mensaje alternativo.
* Fin del criterio de aceptación.

**HU7 – Guardar progreso del jugador**

**Como** jugador,  
**quiero** guardar mi progreso actual,  
**para** poder continuar más adelante desde el mismo punto.

**Criterios de aceptación:**

* El sistema debe guardar los datos en Firebase (nivel, puntaje, decisiones).
* La operación de guardado debe confirmarse visualmente.
* Debe existir persistencia incluso si la app se cierra inesperadamente.

**HU8 – Salir del juego**

**Como** jugador,  
**quiero** salir del juego de forma segura,  
**para** cerrar la aplicación sin perder mi información.

**Criterios de aceptación:**

* Al seleccionar “Salir”, el sistema debe guardar automáticamente el progreso.
* La app debe cerrar la sesión y volver a la pantalla inicial.
* Fin del criterio de aceptación.

**🛠️** Rol: Administrador

**HU9 – Gestión de contenido educativo**

**Como** administrador,  
**quiero** agregar, editar o eliminar preguntas de las trivias,  
**para** mantener actualizado el contenido educativo del juego.

**Criterios de aceptación:**

* El sistema debe permitir crear y modificar preguntas desde una interfaz administrativa.
* Los cambios deben guardarse en la base de datos Firebase.
* Las preguntas deben reflejarse correctamente en el modo de juego.
* Fin del criterio de aceptación.

**HU10 – Monitoreo del progreso de los jugadores**

**Como** administrador,  
**quiero** visualizar el avance de los jugadores,  
**para** evaluar su aprendizaje y rendimiento dentro del juego.

**Criterios de aceptación:**

* El sistema debe mostrar un panel con el listado de jugadores y su progreso.
* El administrador debe poder filtrar por nombre o nivel.
* Fin del criterio de aceptación.

**🤖** Rol: Inteligencia Artificial

**HU11 – Generar contenido visual personalizado**

**Como** sistema de inteligencia artificial,  
**quiero** generar imágenes o escenas según las respuestas del jugador,  
**para** adaptar la historia de manera visual e interactiva.

**Criterios de aceptación:**

* El sistema debe recibir la decisión del jugador como entrada.
* Debe producir un resultado visual coherente (imagen o escena).
* El tiempo de generación no debe exceder los 3 segundos.
* Fin del criterio de aceptación.

**HU12 – Proporcionar retroalimentación educativa**

**Como** sistema de inteligencia artificial,  
**quiero** ofrecer mensajes de refuerzo y corrección,  
**para** apoyar el aprendizaje del jugador de manera personalizada.

**Criterios de aceptación:**

* El sistema debe analizar la respuesta del jugador.
* Debe generar un mensaje o audio educativo.
* Si la IA no responde, debe mostrarse una retroalimentación por defecto.
* Fin del criterio de aceptación.

# 7. Casos de Uso del Sistema

### 7.1 Actores del Sistema

### 7.2 Descripción de Casos de Uso

# 8. Arquitectura de Software

### 8.1 Principios SOLID aplicados

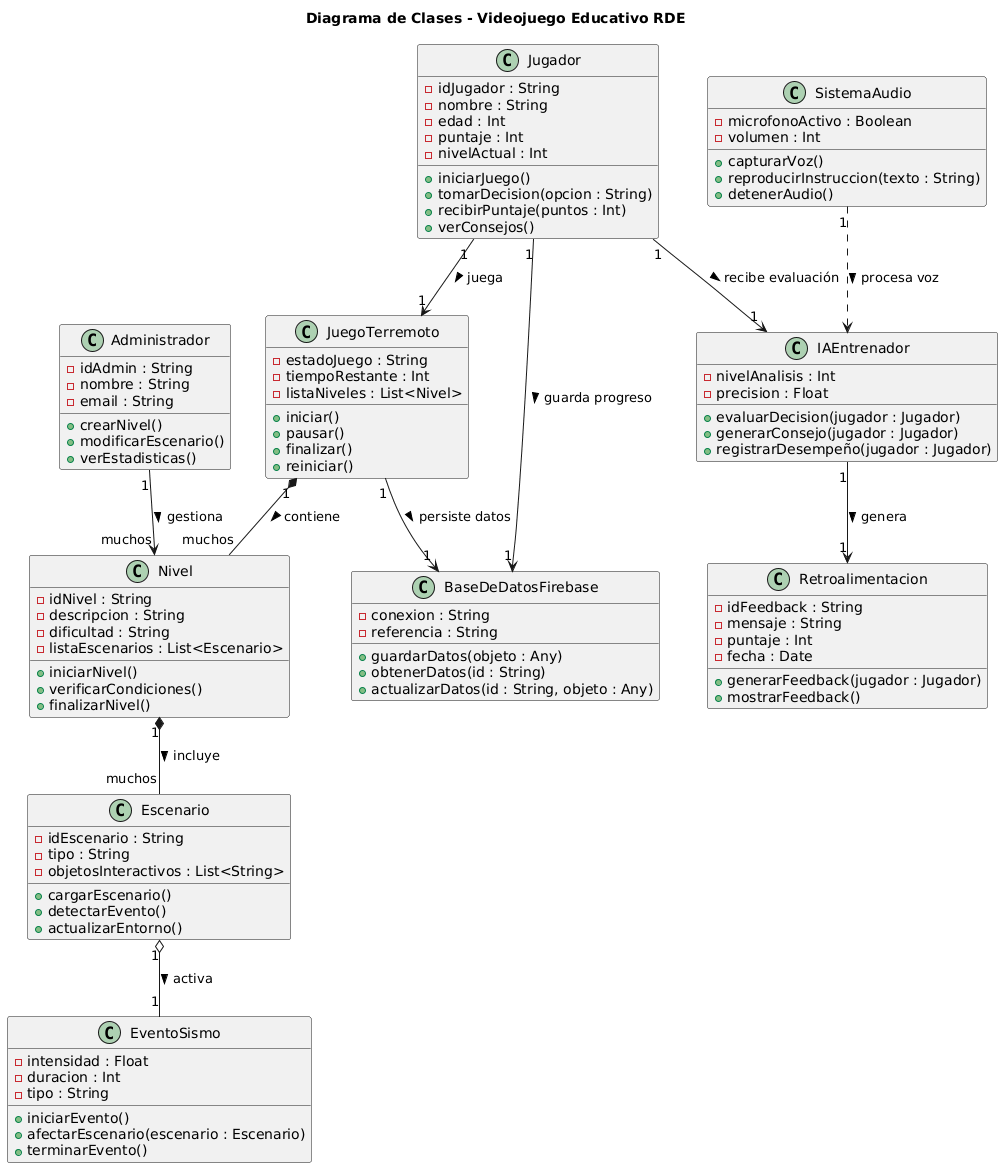
### 8.2 Patrón arquitectónico: MVVM (y mención de Hexagonal como alternativa)

### 8.3 Microservicios utilizados (solo si se integran APIs externas)

### 8.4 Diagramas C4 (Nivel de contexto, contenedores y componentes)

# 9. Modelado Estructural

### 9.1 Diagrama de Clases (UML)



### **9.2 Diagrama de Objetos** (opcional para este alcance)

### 9.3 Diagrama de Componentes

### **9.4 Diagrama de Despliegue** (para mostrar cómo se ejecuta en Android con Firebase)

### **9.5 Diagrama de Paquetes** (opcional, según la arquitectura MVVM)

# 10. Modelado de Comportamiento

### 10.1 Diagramas de Casos de Uso (Jugador, Administrador, Sistema)

### 10.2 Diagrama de Actividad

### 10.3 Diagrama de Estado *(para flujo del jugador o partida)*

### 10.4 Diagrama de Secuencia

# 11. Diagramas de Arquitectura

### 11.1 Diagrama de Arquitectura de Software

### 11.2 Diagrama de Arquitectura del Sistema

### 11.3 Diagrama de Arquitectura de Aplicaciones

### 11.4 Diagrama de Arquitectura de Integración (IA - Firebase)

### 11.5 Diagrama de Arquitectura de Implementación (Kotlin - XML)

# 12. Escenarios Operativos

(Cómo se espera que los usuarios interactúen con la app.)

# 13. Cronograma o Calendario Actualizado

(Fechas, entregables y tareas.)

# 14. Conclusiones

# 15. Bibliografía