

1. Resumen Ejecutivo

1.1 Breve descripción:

El proyecto consiste en una aplicación móvil diseñada para ayudar a jugadores de Mus a aprender, mejorar sus decisiones y comprender mejor las probabilidades del juego. La app permite introducir manos reales, analizar situaciones mediante simulaciones Monte Carlo y recibir recomendaciones claras y justificadas. También incluye un modo de entrenamiento y estadísticas personales para seguir el progreso.

1.2 Objetivos principales:

- Mejorar el nivel estratégico del usuario mediante análisis automatizados.
- Facilitar el aprendizaje del Mus mediante un sistema de entrenamiento guiado.
- Ofrecer estadísticas avanzadas que permitan ver progresos y errores.
- Crear una base tecnológica escalable para futuras versiones.
- Monetizar la aplicación mediante modelo freemium, que consistirá en cuatro gamas del producto, una versión gratuita con anuncios, y luego tres “premium” de pago que serán las siguientes: El plan Básico que no tendrá anuncios, el plan Pro que contendrá simulaciones avanzadas y estadísticas detalladas y el plan Élite con exportación de datos y prioridad de servidor.

1.3 Alcance general:

La aplicación permitirá introducir la mano del jugador y el estado de la partida, analizar probabilidades reales, mostrar recomendaciones óptimas y permitir entrenar con escenarios preparados. El sistema funcionará de manera online mediante una conexión a una base de datos que es la que procesa los datos y tiene el motor monte carlo, centrándose únicamente en el análisis y aprendizaje del Mus, sin incluir multijugador ni reconocimiento automático de cartas.

1.4 Beneficios esperados:

Para los usuarios:

- Aprendizaje accesible y progresivo del Mus.
- Mejor comprensión de las decisiones gracias a explicaciones claras.

- Estadísticas que muestran la evolución del jugador.
- Funcionamiento con Internet.

Para inversores:

- Producto con un enfoque único y alto potencial de crecimiento.
- Explotación de un nicho desatendido por el mercado, siendo nosotros la primera “solución”.
- Modelo de monetización sostenible y consolidado por otras empresas del sector.
- Base escalable para añadir multijugador o IA avanzada en el futuro.

1.5 Cronograma:

El proyecto se desarrollará en aproximadamente 22 semanas:

- Semanas 1–3: Análisis y especificación del sistema.
- Semanas 4–7: Diseño de arquitectura, diagramas y prototipos.
- Semanas 8–15: Desarrollo de módulos principales.
- Semanas 16–19: Pruebas y optimización.
- Semanas 20–21: Documentación y empaquetado.
- Semana 22: Presentación final a inversores.

2. Antecedentes y Justificación

2.1. Contexto del problema o necesidad

El mus es un juego de cartas profundamente arraigado en la cultura española, transmitido de generación en generación tanto en entornos familiares como sociales. A pesar de su amplia popularidad, muchos jugadores lo perciben como un juego especialmente complejo, lo que desincentiva su aprendizaje y práctica. Esta percepción se fundamenta en diversos factores.

En primer lugar, la **falta de información disponible para el jugador** condiciona la toma de decisiones. En una partida con cuatro jugadores y dieciséis cartas repartidas (cuatro por cada jugador), doce de ellas son desconocidas. Esta incertidumbre genera inseguridad al no saber con exactitud si es posible ganar la mano en curso. Aunque existe un sistema de señas utilizado para compartir información con el compañero, en la práctica suele ser insuficiente para resolver por completo esta limitación.

En segundo lugar, el mus combina elementos de **azar, estrategia, comunicación limitada y psicología**, lo que incrementa su complejidad. Aunque el juego presenta constantes numéricas —como el número total de cartas o la distribución fija de figuras—, los jugadores noveles pueden sentirse abrumados al no dominar el conteo, la interpretación de señales o el análisis del comportamiento de rivales y compañero.

En tercer lugar, se observa una **escasez significativa de literatura científica** relacionada con el mus. A diferencia de juegos como el póker, que cuenta con numerosos estudios académicos, programas especializados y un entorno profesionalizado, el mus no ha experimentado este desarrollo teórico. Esta carencia dificulta su análisis estructurado y limita la creación de herramientas avanzadas que contribuyan a su comprensión.

A estos factores se suma la **poca profesionalización del entorno competitivo**. Los torneos de mus suelen carecer de estadísticas oficiales, análisis detallados o sistemas estandarizados de evaluación, especialmente si se comparan con juegos de azar o disciplinas de estrategia más desarrolladas. La incorporación de herramientas matemáticas y modelos probabilísticos podría mejorar la calidad de las competiciones y ofrecer una experiencia más informada tanto para jugadores como para espectadores.

Por último, en un contexto en el que las nuevas tecnologías están transformando la forma de ocio, los juegos tradicionales corren el riesgo de perder relevancia. La digitalización del mus contribuiría a garantizar su continuidad en las próximas décadas. Asimismo, la introducción de inteligencia artificial en este ámbito permitiría acercarlo a nuevas generaciones, haciendo el juego más accesible, atractivo y adaptable a los entornos digitales actuales.

2.2. Situación actual: qué se intenta mejorar o resolver

En la actualidad, aunque el mus sigue siendo practicado en bares, hogares y peñas, su **análisis técnico y formal** continúa siendo muy limitado. La mayor parte del conocimiento deriva de la experiencia personal o de recomendaciones transmitidas oralmente, lo que impide contar con una referencia objetiva sobre la calidad de las decisiones tomadas durante una partida.

Uno de los principales problemas es la **ausencia de modelos matemáticos claros**. A diferencia de juegos como el póker o el blackjack, que disponen de simuladores, estudios probabilísticos y cursos especializados, el mus carece de herramientas que permitan calcular probabilidades fiables o determinar la mejor opción en cada lance. No existe actualmente una forma accesible de conocer la probabilidad real de ganar la grande, la chica o los pares con una mano determinada, lo que hace que el progreso del jugador dependa más de la intuición que de datos objetivos.

Otro aspecto relevante es la **diversidad de reglas y la falta de un estándar oficial**. Las diferencias regionales afectan a las señas permitidas, al tanteo o al valor otorgado a determinadas jugadas. Aunque esta riqueza cultural es parte de la identidad del mus, dificulta el desarrollo de modelos predictivos generalizables. Tal como ocurrió con el póker —que logró consolidar el Texas Hold'em como variante de referencia pese a la existencia de múltiples versiones—, el análisis técnico podría contribuir a establecer una modalidad estándar del mus y clasificar el resto como variantes reconocidas.

Asimismo, el mus incorpora factores psicológicos y sociales, como la interpretación de señas, la lectura del rival y la gestión emocional. Estos elementos, aunque esenciales, son difíciles de analizar sin una base objetiva, lo que complica especialmente el aprendizaje de los jugadores noveles.

En el ámbito competitivo, los torneos suelen carecer de estadísticas, análisis post-partida o herramientas que evalúen el rendimiento de las parejas. Mientras otros juegos han evolucionado hacia entornos más profesionalizados, el mus mantiene prácticas similares a las de décadas pasadas. La introducción de modelos matemáticos y algoritmos predictivos podría mejorar la organización, aportar datos precisos y elevar la calidad del juego competitivo.

Finalmente, aunque existen plataformas digitales para jugar al mus online, la mayoría carece de inteligencias artificiales suficientemente avanzadas como para simular el comportamiento estratégico de un jugador experto. Esta limitación afecta tanto al aprendizaje como al entrenamiento. En un mundo cada vez más digitalizado, la modernización del mus es esencial para garantizar su supervivencia y su relevancia cultural.

2.3. Impacto esperado (económico, técnico y social)

El desarrollo de un algoritmo predictivo orientado al mus no solo busca aportar una perspectiva matemática del juego, sino también generar un impacto significativo en diversos ámbitos: técnico, competitivo, económico y social.

Impacto técnico

La creación de modelos predictivos permitirá obtener información que actualmente resulta inaccesible: probabilidades de éxito en cada lance, valor esperado de las decisiones, simulaciones completas de manos y recomendaciones basadas en datos. Estos avances facilitarán el aprendizaje, elevarán el nivel global de juego y servirán de base para desarrollar aplicaciones, motores de inteligencia artificial y sistemas de análisis avanzados.

Impacto en el ámbito competitivo

La introducción de estadísticas, análisis post-partida y herramientas de seguimiento profesionalizará los torneos. Será posible estudiar patrones, comparar estrategias y crear rankings basados en datos objetivos. Este enfoque acercará el mus al nivel de otros juegos competitivos y permitirá establecer una modalidad estándar, de forma similar a la consolidación del Texas Hold'em en el póker.

Impacto económico

La profesionalización atraerá a patrocinadores, organizadores y empresas interesadas en desarrollar plataformas, aplicaciones móviles, contenidos formativos o eventos retransmitidos. La digitalización del mus puede abrir nuevos mercados, desde academias online hasta competiciones digitales. La existencia de un motor matemático fiable incrementará notablemente la viabilidad y atractivo de estas iniciativas.

Impacto social y cultural

El mus forma parte del patrimonio cultural de numerosas regiones de España, pero su continuidad depende de su capacidad de adaptarse a las nuevas formas de ocio. Incorporar tecnologías como la inteligencia artificial permitirá modernizarlo sin perder su esencia. Además, la digitalización y el análisis formal contribuirán a documentar sus variantes y preservar su historia. Esto facilitará su transmisión a nuevas generaciones y garantizará su presencia en la cultura popular futura.

En conjunto, el mus se encuentra en un punto de transición: conserva su tradición, pero carece de una base técnica sólida que permita estudiarlo, profesionalizarlo y adaptarlo a los entornos digitales actuales. El desarrollo de un algoritmo predictivo representa un paso necesario para modernizar el juego, ampliar su alcance y asegurar su continuidad cultural, competitiva y social.

3. Objetivos del Proyecto

Nuestro propósito es modernizar la transmisión de conocimiento de uno de los juegos de cartas más complejos y tradicionales, transformando la experiencia de aprendizaje mediante tecnología de vanguardia y conectividad.

3.1. Objetivo General

Desarrollar y lanzar al mercado una plataforma online interactiva de aprendizaje de Mus que, mediante la integración de un motor de inteligencia artificial basado en simulaciones Monte Carlo en la nube, permita a jugadores de cualquier nivel perfeccionar su técnica, analizar jugadas en tiempo real y comprender la lógica matemática detrás de cada decisión, digitalizando la experiencia de un mentor experto.

3.2. Objetivos Específicos

Para garantizar el cumplimiento de nuestra visión y la viabilidad del modelo de negocio, hemos desglosado el proyecto en las siguientes metas concretas y medibles:

- **Implementar el Motor de Decisión Remoto (Core):** Desarrollar un algoritmo de recomendación basado en el método de Monte Carlo alojado en servidores externos, capaz de procesar miles de simulaciones por petición y devolver la mejor opción estadística (envite, órdago o paso) al dispositivo del usuario con una latencia inferior a 3 segundos.
- **Optimizar la Experiencia de Usuario (Input):** Diseñar y validar una interfaz de entrada manual de cartas y estados de partida (tanteo, mano, postre) que sea intuitiva y rápida, permitiendo al usuario configurar una situación de juego real en menos de 15 segundos para no interrumpir el flujo de aprendizaje .
- **Desarrollar el Módulo de Entrenamiento Adaptativo:** Crear un sistema de escenarios predefinidos escalables en 3 niveles de dificultad (Principiante, Intermedio, Experto) que cubran las fases críticas del juego, proporcionando retroalimentación didáctica basada en los aciertos y errores del usuario.
- **Integrar Analítica de Rendimiento en la Nube:** Construir un panel de estadísticas persistente vinculado a un perfil de usuario registrado, que almacene el historial de decisiones y permita visualizar la evolución del aprendizaje y la tasa de acierto respecto a la IA desde cualquier dispositivo .
- **Validar el Modelo de Negocio (Freemium):**

Lanzamiento del MVP (Producto Mínimo Viable): Desplegar una versión funcional y estable de la aplicación (excluyendo multijugador en tiempo real) lista para pruebas de usuario y demostración a inversores en un plazo de 22 semanas, cumpliendo con los estándares de calidad definidos en el plan de gestión.

4. Alcance del Proyecto

El presente apartado define los límites, funcionalidades, entregables y criterios de aceptación de la primera versión de la aplicación, orientada al análisis avanzado, entrenamiento y aprendizaje del Mus.

El sistema consistirá en una aplicación de escritorio completamente offline especializada en el aprendizaje del Mus, con herramientas de análisis inteligentes y módulos interactivos orientados tanto a jugadores principiantes como a usuarios experimentados que deseen mejorar su nivel estratégico.

Además, la aplicación se distribuirá bajo un modelo de cuatro planes: Plan gratuito, Plan Básico, Plan Pro, Plan Elite.

4.1. Funcionalidades que se incluirán

4.1.0 Modos de producto:

Plan Gratuito

ofrece una versión limitada pero plenamente funcional de la aplicación, orientada a usuarios que deseen explorar el sistema antes de adquirir una suscripción superior.

Plan Básico:

Incluye las funcionalidades esenciales del sistema, sin publicidad y con acceso al núcleo del análisis, manteniendo las simulaciones estándar y el uso básico del modo de entrenamiento.

Plan Pro:

Incluye todas las características del Plan Básico **más**: simulaciones avanzadas, estadísticas ampliadas y análisis adicionales y ajustes específicos del motor de recomendación.

Plan Elite:

Incluye todo lo anterior **más**: **exportación de datos** en múltiples formatos (PDF, CSV, etc.) sin restricciones y funcionalidades extendidas que requieren mayor capacidad de análisis.

4.1.1. Módulo de Introducción de Manos y Estados de Partida

Este módulo constituye uno de los ejes principales del sistema, ya que permite al usuario reproducir situaciones reales de juego para analizar decisiones óptimas.

Incluye:

- Introducción manual de las cuatro cartas del jugador, mediante selección visual de la baraja o entrada por texto.
- Configuración del estado de la mano, incluyendo:
 - Mano en curso (grande, chica, pares, juego).
 - Situación de apuestas (envido, órdago, paso, etc.).
 - Señales permitidas (si el usuario quiere entrenar este aspecto opcionalmente).
 - Número de puntos acumulados por cada pareja.
 - Ronda en curso (mano o postre).
- Guardado de configuraciones de partida para su reutilización posterior.
- Validación automática para evitar inconsistencias (ej.: combinaciones inválidas, estados imposibles, duplicados de cartas).

Este módulo permitirá al usuario simular desde simples manos iniciales hasta momentos críticos del juego.

4.1.2. Motor de Recomendación

Constituye el principal atractivo para los inversores por su aporte diferenciador.

El sistema incluirá:

- Ejecución automática de miles o millones de simulaciones por escenario.
- Modelado probabilístico de manos posibles del resto de jugadores.
- Evaluación de:
 - Probabilidades de victoria según la fase.
 - Riesgo asociado a cada decisión.
 - Jugadas más rentables a largo plazo.
- Generación de una recomendación principal y varias alternativas viables.
- Justificación textual y visual de por qué esa jugada es la recomendada.
- Tres niveles de análisis:
 - Básico: simulación simplificada, ideal para principiantes.
 - Intermedio: análisis equilibrado.

- Avanzado: simulación exhaustiva para usuarios expertos.

El motor se diseñará para ser modular, escalable y optimizado, garantizando tiempos de respuesta adecuados.

4.1.3. Modo de Entrenamiento Interactivo

El sistema ofrecerá un entorno guiado que permitirá al usuario aprender progresivamente las estrategias fundamentales del Mus.

Incluye:

- Escenarios predefinidos por nivel de dificultad, diseñados en colaboración con expertos y jugadores reales.
- Casuísticas típicas de partida (falta de reyes, juegos débiles, pares fuertes, manos engañosas, etc.).
- Explicaciones didácticas adaptadas al nivel del usuario.
- Retroalimentación inmediata tras cada decisión, incluyendo:
 - Alternativas mejores.
 - Porcentaje de mejora o de riesgo evitado.
 - Consejos estratégicos generales.
- Sistema de progreso y logros para motivar el aprendizaje continuo.

4.1.4. Panel de Estadísticas y Análisis del Rendimiento

El software incorporará un centro de análisis personal basado en el historial del usuario.

- Registro automático de:
 - Escenarios analizados.
 - Decisiones tomadas.
 - Jugadas recomendadas ignoradas o aplicadas.
 - Nivel de riesgo promedio.
 - Tendencias en cada fase del Mus.
- Visualización mediante gráficos:
 - Porcentaje de decisiones correctas.

- Evolución temporal.
- Categorías de fallos recurrentes.
- Comparativa entre niveles de dificultad.
- Sistema de autoevaluación guiado:
 - “Puntos fuertes”.
 - “Aspectos a mejorar”.
 - “Recomendaciones personalizadas”.
- Exportación en PDF o CSV para análisis externo o consulta posterior.

4.1.5. Interfaz de Usuario y Experiencia de Uso

La aplicación incluirá:

- Diseño centrado en la usabilidad y la claridad didáctica.
- Representación visual de cartas, manos y estados de juego.
- Navegación sencilla y estructurada en módulos.
- Modo oscuro y accesibilidad básica.
- Pantalla inicial de inicio de sesión para acceder al sistema.

4.1.6. Datos Locales y Persistencia

- Sistema de guardado de escenarios creados por el usuario.
- Preferencias de configuración.
- Historial de simulaciones realizadas.
- Estadísticas personales.

4.2. Límites del Sistema

4.2. Límites del Sistema (No incluido en esta versión)

Para mantener la viabilidad del proyecto, asegurar la calidad del producto y garantizar su entrega dentro de los plazos establecidos, se definen explícitamente las funcionalidades, características y servicios que quedan fuera del alcance de la primera versión.

Estas exclusiones permiten concentrar los recursos del desarrollo en los módulos principales del sistema evitando desviaciones en tiempo, coste o complejidad.

Las siguientes capacidades no serán implementadas en esta fase, ya sea por su complejidad, por no ser esenciales para el objetivo de esta versión o por requerir arquitectura adicional no contemplada:

- **Modo multijugador en tiempo real o por turnos:**
No se incluirá ningún tipo de interacción entre distintos usuarios dentro de la plataforma.
- **Conexión con otros jugadores, salas, matchmaking o torneos:**
No existirán mecanismos para organizar partidas, crear salas virtuales, realizar emparejamientos automáticos ni participar en competiciones estructuradas.
- **Integración con redes sociales o comunidades online:**
No se contempla la posibilidad de compartir partidas, estadísticas, logros o resultados en plataformas externas.
- **Reconocimiento automático de cartas mediante cámara o visión artificial:**
La aplicación no realizará lectura de cartas reales captadas por dispositivos móviles o cámaras externas; toda entrada deberá ser manual.
- **Implementación de variantes regionales del Mus:**
Solo se aplicarán las reglas estándar del juego. Versiones locales (Navarra, Aragonés, Vasco, Castellano, etc.) quedan fuera de esta versión.
- **Gamificación avanzada** (misiones diarias, recompensas virtuales, rankings):
Aunque el sistema incluirá un progreso básico y logros simples, no se desarrollarán mecánicas avanzadas de gamificación.

4.3. Entregables Principales

Al finalizar el proyecto, se entregarán los siguientes artefactos:

4.3.1. Entregables de análisis y diseño

- Documento SRS completo.
- Documento de arquitectura del sistema.
- Casos de uso, diagramas UML, diagramas de actividad.
- Prototipado de interfaz.

4.3.2. Entregables de desarrollo

- Implementación del módulo de entrada de escenarios.

- Sistema de recomendación con explicación.
- Modo de entrenamiento interactivo.
- Panel de estadísticas.
- Persistencia local y sistema de guardado.

4.3.3. Entregables de validación y soporte

- Plan de pruebas completo (unitarias, integración, sistema).
- Informe de resultados de pruebas.
- Manual de usuario.
- Manual técnico para desarrolladores.
- Versión ejecutable y paquete instalable.

4.4. Criterios de Aceptación

Para considerar que el proyecto se ha completado con éxito y que la versión entregada cumple los requisitos establecidos, es necesario verificar que el sistema satisface todos los criterios funcionales, de rendimiento, de usabilidad y de calidad definidos en este apartado. Estos criterios servirán como referencia para la validación final del producto, garantizando que la aplicación opera de manera correcta, estable y conforme a las expectativas del usuario y del equipo de desarrollo.

4.4.1. Criterios funcionales

- El usuario puede introducir una mano y un estado de partida válidos sin errores.
- El motor genera una recomendación en un tiempo razonable.
- El modo de entrenamiento presenta escenarios correctos y ofrece retroalimentación.
- Las estadísticas se actualizan automáticamente y reflejan los datos reales del usuario.
- El sistema funciona de manera completamente local.

4.4.2. Criterios de rendimiento

- Tiempo de recomendación inferior a un límite fijado (por ejemplo: <3 segundos por análisis estándar).
- La aplicación responde sin bloqueos o congelamientos.
- El consumo de recursos está dentro de los márgenes definidos.

4.4.3. Criterios de usabilidad

- La interfaz permite que un usuario sin experiencia previa utilice el modo de entrenamiento sin guía externa.
- Las explicaciones del motor son comprensibles y aportan valor al aprendizaje.
- La navegación es clara y consistente.

4.4.4. Criterios de calidad

- Se superan todas las pruebas definidas en el plan de validación.
- No existen errores críticos abiertos en la versión final.

La documentación técnica y de usuario está completa y correctamente presentada.

5. Requerimientos del Sistema

En este apartado se definen de forma detallada los requerimientos funcionales, no funcionales y de usuario de la aplicación móvil de análisis y aprendizaje del Mus. Estos requisitos especifican qué debe hacer exactamente el sistema, bajo qué condiciones debe operar y qué expectativas tienen los usuarios finales. Además, se integran los mecanismos de autenticación, monetización mediante pasarela de pago y los distintos planes comerciales disponibles dentro de la aplicación.

Este apartado es uno de los más críticos del proyecto, ya que conecta directamente los objetivos educativos con el modelo de negocio freemium y la viabilidad económica para los inversores.

5.1. Requerimientos funcionales

Los requerimientos funcionales describen las acciones y comportamientos que el sistema debe ser capaz de realizar desde el punto de vista operativo.

5.1.1. Gestión de la mano del jugador

El sistema debe permitir al usuario introducir de forma manual las cuatro cartas que componen su mano mediante una interfaz gráfica basada en la baraja española. La selección se realizará de forma táctil, mostrando claramente los palos y valores para evitar errores de interpretación.

No se permitirá seleccionar cartas duplicadas ni combinaciones imposibles según las reglas del Mus. En caso de error, el sistema mostrará un aviso al usuario indicando el problema detectado.

El usuario podrá editar cualquier carta previamente seleccionada sin necesidad de reiniciar todo el proceso. Asimismo, deberá existir una opción de reinicio rápido que permita borrar completamente la mano en un solo paso.

Como funcionalidad adicional orientada al aprendizaje, el sistema permitirá la generación automática de manos aleatorias con fines de práctica.

5.1.2. Introducción y validación del estado de la partida

El sistema permitirá introducir el contexto completo de la partida, necesario para la correcta ejecución de las simulaciones.

El usuario deberá indicar en qué fase del Mus se encuentra (grande, chica, pares o juego). Además, podrá introducir el marcador actual de la partida indicando los tantos acumulados por cada pareja.

El sistema permitirá registrar si existe un envite activo, su cuantía y si se han producido subidas previas o un órdago. Asimismo, se indicará si ha habido mus y cuántas cartas se han descartado.

Todos los datos introducidos serán validados automáticamente por el sistema para garantizar su coherencia con las reglas del juego. Ante cualquier incoherencia, se mostrará un mensaje explicativo indicando cómo solucionarla.

5.1.3. Motor de simulación mediante Monte Carlo

El sistema deberá contar con un motor de simulación probabilística capaz de generar miles de escenarios virtuales a partir de la información introducida por el usuario.

A partir de las cartas conocidas, el sistema generará repartos compatibles de las cartas restantes entre los jugadores rivales, respetando las restricciones impuestas por el mazo.

El sistema ejecutará simulaciones masivas de partidas para calcular:

- Probabilidad real de ganar cada lance.
- Valor esperado (EV) de cada acción posible.
- Nivel de riesgo asociado a cada jugada.

Estos cálculos permitirán ofrecer recomendaciones basadas en datos objetivos y estadísticamente sólidos.

5.1.4. Sistema de recomendaciones inteligentes

El sistema mostrará al usuario la jugada óptima en función de los resultados de las simulaciones realizadas.

Cada recomendación irá acompañada de una explicación en lenguaje natural, adaptada al nivel de conocimiento del usuario.

El sistema permitirá comparar la jugada recomendada con otras alternativas, mostrando las diferencias en probabilidad de éxito y riesgo.

Además, se incluirán indicadores visuales como porcentajes, barras de probabilidad y niveles de riesgo para facilitar la interpretación de los resultados.

5.1.5. Modo de entrenamiento guiado

El sistema incorporará un modo de entrenamiento con tres niveles de dificultad: principiante, intermedio y avanzado.

Cada nivel contará con escenarios predefinidos diseñados específicamente para entrenar conceptos concretos del Mus.

El sistema evaluará automáticamente cada decisión tomada por el usuario, clasificándola como correcta, aceptable o errónea.

Tras cada ejercicio, se mostrará una explicación detallada de la decisión correcta para fomentar el aprendizaje progresivo.

5.1.6. Panel de estadísticas personales

El sistema almacenará el historial completo de manos analizadas, decisiones tomadas y recomendaciones recibidas.

A partir de estos datos se generarán estadísticas de rendimiento por fase del juego.

Se mostrará la evolución temporal del usuario mediante gráficos y resúmenes.

Las estadísticas estarán parcialmente limitadas en la versión gratuita y completamente disponibles en las versiones de pago.

5.1.7. Registro, autenticación y gestión de usuarios

El sistema deberá permitir el registro de usuarios mediante correo electrónico y contraseña.

El sistema exigirá el inicio de sesión para poder acceder a las funcionalidades completas de la aplicación.

Las contraseñas se almacenarán de forma cifrada.

Cada usuario dispondrá de un perfil personal donde se gestionarán sus datos, preferencias y tipo de suscripción.

El sistema permitirá un modo invitado con acceso muy limitado a ciertas funcionalidades básicas sin registro.

5.1.8. Pasarela de pago integrada

El sistema deberá integrar una pasarela de pago segura que permita realizar compras dentro de la aplicación.

Se deberán soportar métodos de pago digitales seguros compatibles con dispositivos móviles.

El sistema validará automáticamente cada transacción antes de desbloquear cualquier contenido de pago.

Se registrará un historial de compras asociado a cada usuario.

En caso de error en el pago, el sistema informará al usuario de forma clara y segura.

5.1.9. Gestión de productos y planes comerciales

El sistema deberá gestionar tres tipos de productos digitales basados en planes de suscripción:

Plan Básico

Permitirá eliminar por completo la publicidad dentro de la aplicación, mejorando la experiencia del usuario.

Plan Pro

Incluirá acceso a simulaciones avanzadas, mayor profundidad de análisis y estadísticas detalladas sobre el rendimiento del jugador.

Plan Elite

Permitirá la exportación de datos, acceso prioritario al servidor y máxima velocidad en las simulaciones.

El sistema deberá permitir la compra, renovación y cancelación de estos planes.

5.1.10. Funcionamiento online de la aplicación

La aplicación funcionará de manera completamente online, siendo obligatoria la conexión a Internet para ejecutar simulaciones y gestionar datos.

Las simulaciones se realizarán en servidores remotos para no sobrecargar el dispositivo del usuario.

No se incluirá ningún sistema de reconocimiento automático de cartas, siendo obligatoria su introducción manual.

5.2. Requerimientos no funcionales

Los requerimientos no funcionales definen las condiciones de calidad, seguridad, rendimiento y escalabilidad del sistema.

5.2.1. Rendimiento

El tiempo de respuesta estándar de las simulaciones no deberá superar los tres segundos en condiciones normales.

Las simulaciones avanzadas asociadas a los planes Pro y Elite no deberán superar los seis segundos.

El sistema deberá funcionar correctamente en dispositivos móviles de gama media sin bloqueos ni ralentizaciones graves.

5.2.2. Seguridad

Las contraseñas se almacenarán cifradas mediante algoritmos seguros.

Los datos personales de los usuarios estarán protegidos conforme a la normativa vigente de protección de datos.

Las transacciones económicas se realizarán exclusivamente mediante pasarelas de pago certificadas.

El sistema contará con mecanismos de autenticación segura y protección frente a accesos no autorizados.

5.2.3. Usabilidad

La interfaz será clara, intuitiva y adaptada a pantallas táctiles.

El lenguaje utilizado será comprensible incluso para usuarios sin experiencia previa.

Se incorporarán mensajes de ayuda y avisos explicativos.

5.2.4. Portabilidad

La aplicación será compatible con sistemas Android e iOS.

Se adaptará correctamente a diferentes tamaños de pantalla y resoluciones.

5.2.5. Escalabilidad

El sistema estará preparado para futuras ampliaciones como multijugador online, torneos digitales o incorporación de nuevas inteligencias artificiales.

5.2.6. Fiabilidad

El sistema deberá garantizar la persistencia de los datos ante fallos de red o cierres inesperados.

No se permitirá la pérdida de estadísticas ni del historial del usuario.

5.3. Requerimientos de usuario

Los requerimientos de usuario recogen las expectativas, necesidades y percepciones de los distintos perfiles de usuarios finales.

El usuario principiante espera aprender el juego desde cero mediante explicaciones claras, sin necesidad de conocimientos matemáticos avanzados.

El usuario intermedio desea mejorar su nivel estratégico, analizar partidas reales y conocer probabilidades concretas.

El usuario avanzado busca acceso completo a estadísticas, exportación de datos y máxima precisión en las simulaciones.

Todos los usuarios esperan una aplicación estable, fiable, rápida, con una interfaz agradable, un aprendizaje progresivo y resultados estadísticamente sólidos.

6. Arquitectura y Diseño del Sistema

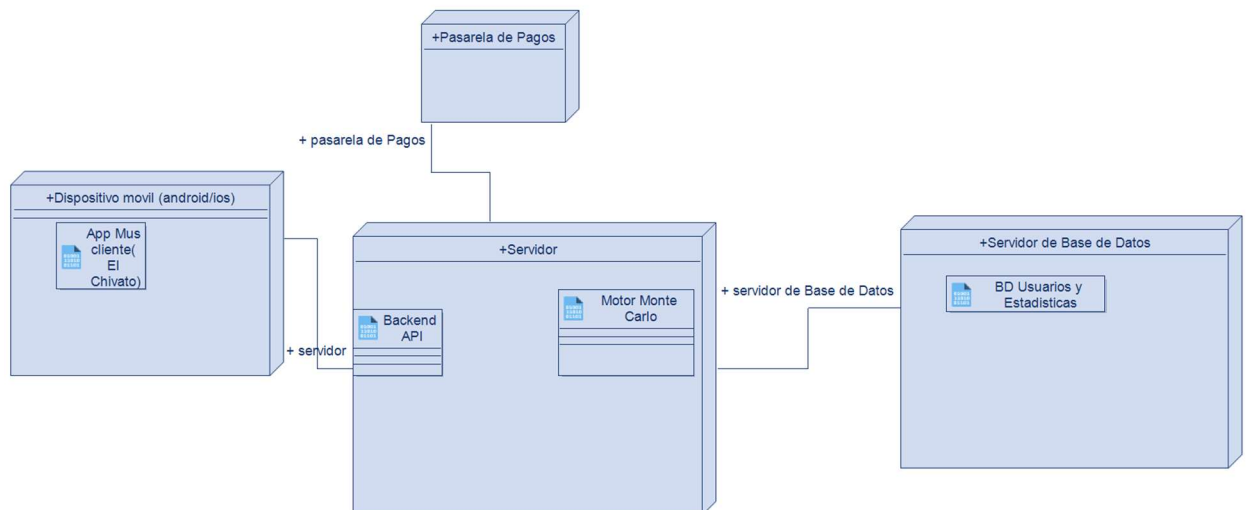
Tras haber establecido en el apartado anterior los requerimientos funcionales y no funcionales del sistema, en este punto se detalla la solución técnica diseñada para darles cumplimiento. El objetivo principal de esta fase es transformar las necesidades de análisis probabilístico y entrenamiento del juego del Mus en una estructura de software robusta, escalable y mantenible.

Dada la naturaleza del proyecto, que requiere la ejecución intensiva de un gran volumen de simulaciones Monte Carlo, se ha optado por una arquitectura distribuida que desacopla la lógica de presentación (dispositivo móvil) de la lógica de negocio y cálculo (servidor). Con esto se responde directamente a la necesidad de garantizar un rendimiento óptimo sin sobrecargar los recursos del usuario final, asegurando tiempos de respuesta adecuados tanto para el análisis de manos como para el modo entrenamiento.

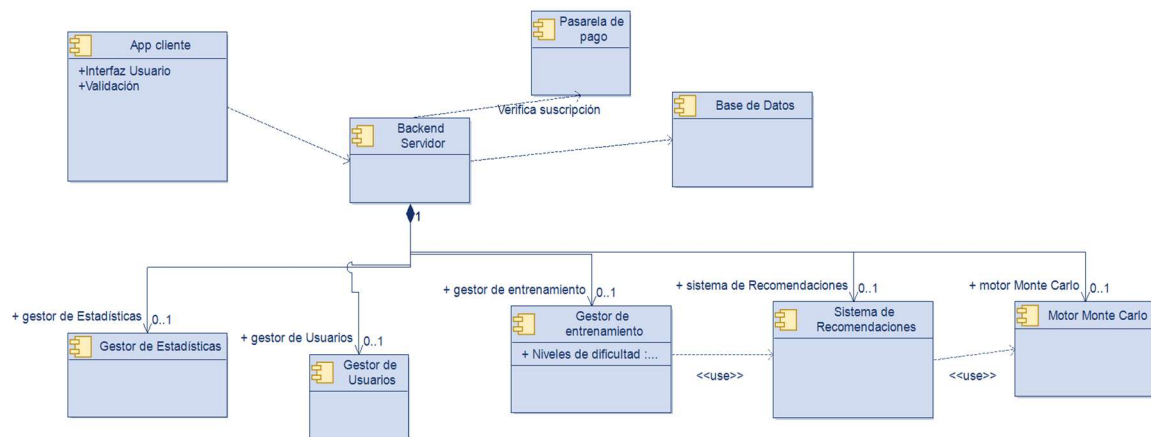
A continuación, se presenta el diagrama general de la arquitectura, seguido de la descomposición en subsistemas lógicos, especificando sus interacciones, flujos de datos y los casos de uso que Implementan.

6.1 Diagrama general de la arquitectura

-Arquitectura física:

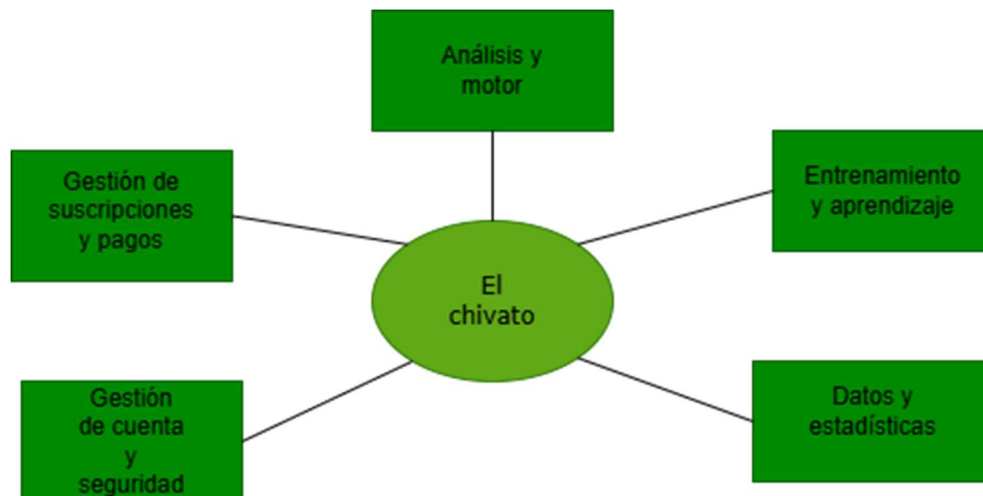


-Arquitectura lógica



-Subsistemas

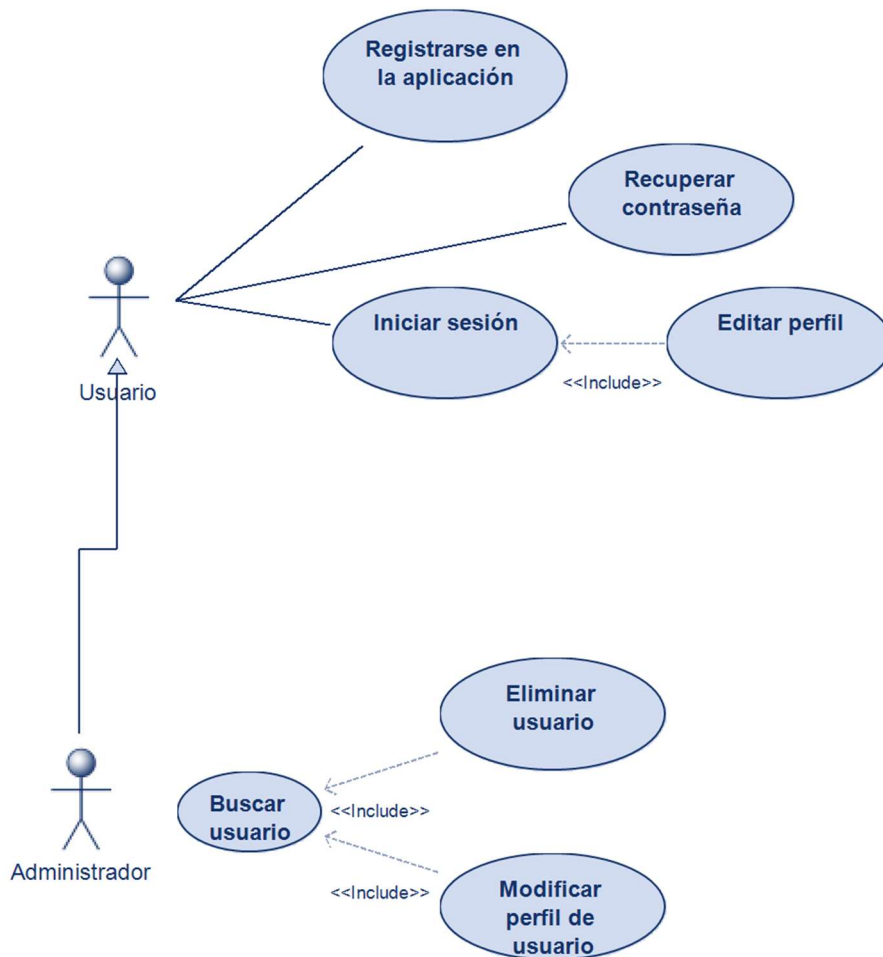
Nuestro sistema se compone principalmente de 5 subsistemas:



A continuación se definirán los subsistemas mencionados.

6.2.1 Gestión de cuenta y seguridad

-Diagrama de casos de uso:



-Especificación de casos de uso:

CASO DE USO #1	Registrarse en la aplicación
Objetivo en contexto	Permitir que un visitante cree una cuenta nueva para acceder a los servicios de la aplicación.
Entradas	Nombre, Apellidos, Email, Nombre de usuario, Contraseña.

CASO DE USO #1	Registrarse en la aplicación
Precondiciones	El usuario no debe estar registrado en el sistema.
Salidas	Mensaje de éxito y redirección a iniciar sesión.
Postcondición si éxito	Se crea un nuevo registro en la base de datos con estado "Activo".
Postcondición si fallo	No se guarda información y se muestran los errores de validación.
Actores	Usuario (Visitante).
Secuencia normal	Paso Acción
	1- El usuario selecciona la opción "Crear cuenta" en la pantalla de inicio.
	2- El sistema muestra el formulario de registro (nombre, email, usuario, contraseña).
	3- El usuario completa los campos obligatorios y pulsa "Registrar".
	4- El sistema valida que el email no esté duplicado y que la contraseña sea segura.
	5- El sistema crea el nuevo registro en la base de datos.
	6- El sistema pide iniciar sesión y muestra la pantalla principal.
Secuencias alternativas	Paso Acción

CASO DE USO #1	Registrarse en la aplicación
	3ª- Cancelar: El usuario pulsa "Cancelar" o "Atrás". El sistema vuelve a la pantalla de inicio sin guardar nada.
	4a- Email ya registrado: El sistema detecta que el correo ya existe. Muestra el error "El correo ya está en uso" y permite al usuario corregirlo o ir a iniciar sesión.
	4b Contraseña débil: El sistema indica que la contraseña no cumple los requisitos (longitud, caracteres, etc.) y pide una nueva.

CASO DE USO #2	Iniciar sesión
Objetivo en contexto	Autenticar a un usuario para permitirle el acceso a funciones privadas.
Entradas	Credenciales (Usuario/Email y Contraseña).
Precondiciones	El usuario debe estar registrado.
Salidas	Acceso al menú principal.
Postcondición si éxito	El sistema inicia una sesión segura para el actor.
Postcondición si fallo	El actor permanece en la pantalla de iniciar sesión con mensaje de error.
Actores	Usuario, Administrador.
Secuencia normal	Paso Acción

CASO DE USO #2	Iniciar sesión
	1- El usuario (o Administrador) introduce su nombre de usuario/email y contraseña.
	2- El usuario pulsa el botón "Entrar".
	3- El sistema valida las credenciales contra la base de datos.
	4- El sistema verifica el rol del usuario (Administrador o Estándar).
	5- El sistema redirige al panel correspondiente según su rol.
Secuencias alternativas	Paso Acción
	3 ^a - Credenciales incorrectas: El sistema no encuentra coincidencia. Muestra "Usuario o contraseña incorrectos" y limpia el campo de contraseña.
	3b- Cuenta bloqueada: El sistema verifica que el usuario está marcado como "Inactivo" o "Baneado". Muestra un mensaje de contacto con soporte.

CASO DE USO #3	Recuperar contraseña
Objetivo en contexto	Permitir restablecer el acceso a la cuenta mediante una validación por correo externo.
Entradas	Correo electrónico.

CASO DE USO #3	Recuperar contraseña
Precondiciones	Ninguna (acceso público).
Salidas	Email con enlace de recuperación.
Postcondición si éxito	Se genera un token temporal de recuperación asociado a la cuenta.
Postcondición si fallo	Mensaje de error (o mensaje neutro por seguridad).
Actores	Usuario.
Secuencia normal	Paso Acción
	1- El usuario selecciona "¿Olvidaste tu contraseña?" en la pantalla de iniciar sesión.
	2- El sistema solicita el correo electrónico asociado a la cuenta.
	3- El usuario introduce el correo y pulsa "Enviar".
	4- El sistema comprueba que el correo existe en la base de datos.
	5- El sistema genera un token de recuperación y envía un email con las instrucciones.
	6- El sistema muestra un mensaje: "Si el correo es correcto, recibirás instrucciones en breve".
Secuencias alternativas	Paso Acción

CASO DE USO #3	Recuperar contraseña
	4ª- Correo no existe: El sistema no encuentra el correo. Muestra el mensaje: “Los datos introducidos no coinciden con ninguno de nuestros usuarios”.
	5ª- Fallo del servidor de correo: El sistema no puede enviar el email por error técnico. Muestra "Error al enviar correo, intente más tarde".

CASO DE USO #4	Editar perfil
Objetivo en contexto	Permitir al usuario modificar sus propios datos personales.
Entradas	Nuevos datos (Foto, dirección, teléfono).
Precondiciones	El usuario ha iniciado sesión.
Salidas	Perfil actualizado.
Postcondición si éxito	Los datos se actualizan en la base de datos.
Postcondición si fallo	Los datos originales se mantienen.
Actores	Usuario.
Secuencia normal	Paso Acción
	1- El usuario accede a la sección "Mi Perfil" y pulsa "Editar".
	2- El sistema carga los datos actuales en el formulario.

CASO DE USO #4	Editar perfil
	3- El usuario modifica los campos deseados (ej: teléfono, dirección) y guarda.
	4- El sistema valida el formato de los nuevos datos.
	5- El sistema actualiza el registro en la base de datos y confirma el éxito.
Secuencias alternativas	Paso Acción
	4 ^a - Datos inválidos: El usuario introduce un formato erróneo (ej. letras en un campo numérico). El sistema resalta el error y no guarda.
	5 ^a - Error de conexión: El sistema pierde conexión con la base de datos. Muestra "No se pudieron guardar los cambios".

CASO DE USO #5	Buscar usuario
Objetivo en contexto	Localizar un usuario específico dentro del sistema.
Entradas	Criterio de búsqueda (ID, nombre, email).
Precondiciones	Ser invocado por "Eliminar usuario" o "Modificar perfil Administrador".
Salidas	Lista de resultados o registro único.
Postcondición si éxito	Se devuelve el objeto "Usuario" seleccionado.

CASO DE USO #5	Buscar usuario
Postcondición si fallo	Retorna lista vacía.
Actores	Sistema (invocado por Administrador).
Secuencia normal	Paso Acción
	1- El caso de uso principal (Eliminar/Modificar) invoca la búsqueda.
	2- El sistema muestra una barra de búsqueda o listado de usuarios.
	3- El Administrador introduce un criterio (Nombre, DNI, Email).
	4-El sistema filtra la base de datos según el criterio.
	5-El sistema muestra los resultados coincidentes para que el Administrador seleccione uno.
Secuencias alternativas	Paso Acción
	5ª- Sin resultados: El sistema informa "No se encontraron usuarios con ese criterio". El Administrador puede intentar otra búsqueda.

CASO DE USO #6	Eliminar usuario
Objetivo en contexto	Dar de baja un usuario del sistema permanentemente.

CASO DE USO #6	Eliminar usuario
Entradas	Selección del usuario a borrar.
Precondiciones	Administrador debe haber iniciado sesión.
Salidas	Confirmación de eliminación.
Postcondición si éxito	El usuario ya no existe o no puede acceder.
Postcondición si fallo	El estado del usuario no cambia.
Actores	Administrador.
Secuencia normal	Paso Acción
	1-El Administrador accede al módulo de gestión de usuarios.
	2-Se ejecuta la secuencia de "Buscar usuario" hasta seleccionar uno. (Es el include)
	3-El Administrador selecciona el usuario y pulsa "Eliminar".
	4-El sistema solicita confirmación explícita ("¿Seguro que desea eliminar a X?").
	5-El Administrador confirma la acción.
	6-El sistema borra el registro (o lo marca como eliminado) en la base de datos.
	7-El sistema actualiza la lista y notifica la eliminación.

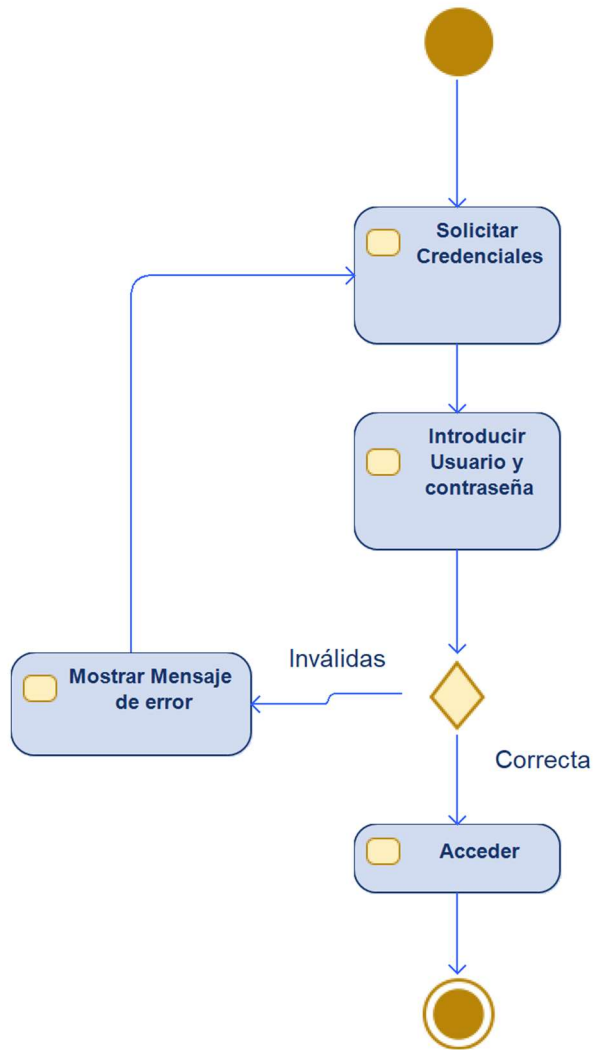
CASO DE USO #6	Eliminar usuario
Secuencias alternativas	Paso Acción
	5ª- Cancelar: El Administrador responde "No" a la confirmación. La operación se aborta y el usuario no se borra.
	6ª- Usuario en uso: Si el usuario tiene datos críticos asociados que impiden el borrado, el sistema muestra error de integridad referencial

CASO DE USO #7	Modificar perfil de usuario (Administrador)
Objetivo en contexto	Permitir al administrador cambiar datos sensibles o roles.
Entradas	Nuevos roles, estado o datos corregidos.
Precondiciones	Administrador debe haber iniciado sesión.
Salidas	Datos del usuario actualizados.
Postcondición si éxito	El usuario afectado tiene nuevos permisos/datos inmediatamente.
Postcondición si fallo	Sin cambios.
Actores	Administrador.
Secuencia normal	Paso Acción
	1-El Administrador accede al módulo de gestión.

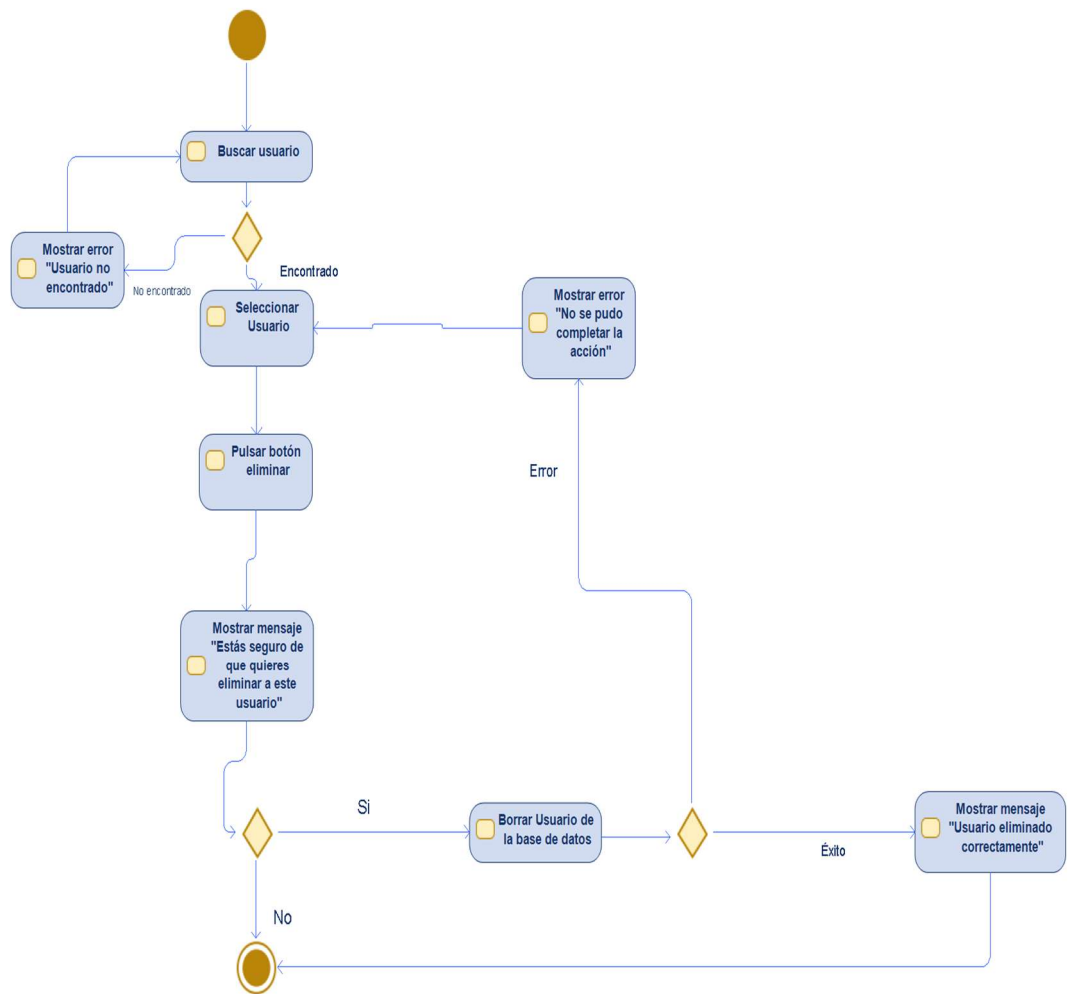
CASO DE USO #7	Modificar perfil de usuario (Administrador)
	2-Se ejecuta la secuencia de "Buscar usuario" hasta seleccionar uno.
	3-El Administrador abre la ficha del usuario para editar.
	4-El Administrador cambia datos sensibles (ej: Asignar rol de 'Editor', desactivar cuenta)
	5-El sistema valida los cambios y guarda en la base de datos.
	6-El sistema confirma "Usuario actualizado correctamente".
Secuencias alternativas	Paso Acción
	5ª- Conflicto de Roles: El Administrador intenta quitar el rol de Administrador al último administrador del sistema. El sistema lo impide para no dejar al mismo sin administradores.
	5b- Datos inválidos: El Administrador introduce un email que ya pertenece a otro usuario. El sistema muestra error de duplicidad.

-Diagramas de actividad

-Inicio de sesión:

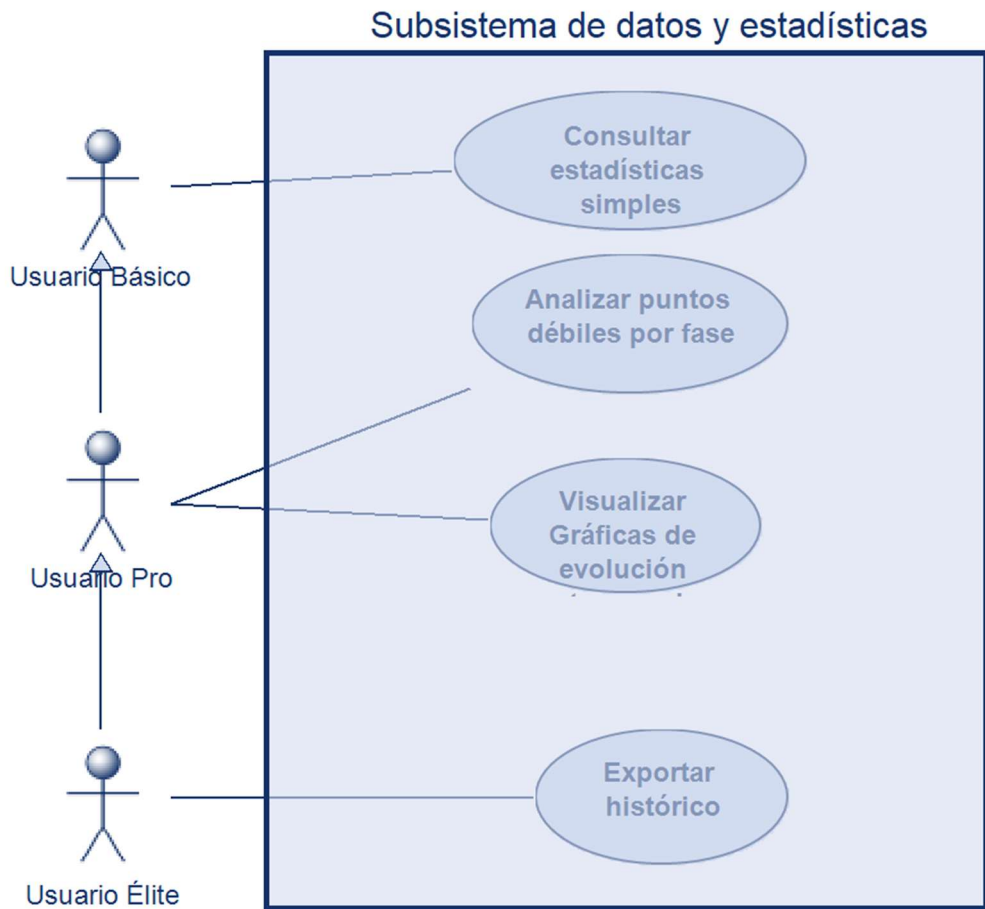


-Eliminar usuario (Administrador)



6.2.2-Subsistema de datos y estadísticas.

-Diagrama de casos de uso



-Casos de uso especificados

CASO DE USO #1	Consultar Estadísticas Simples
Objetivo en contexto	Permitir al usuario conocer su rendimiento global básico (porcentaje de victorias vs derrotas).
Entradas	Selección de periodo (opcional: Hoy, Total).
Precondiciones	El usuario debe haber iniciado sesión y haber jugado al menos un escenario.
Salidas	Gráficos circulares y contadores de Aciertos/Fallos.

CASO DE USO #1	Consultar Estadísticas Simples
Postcondición si éxito	Se visualiza la información actualizada en pantalla.
Postcondición si fallo	Se muestra mensaje indicando que no hay datos suficientes.
Actores	Usuario (Básico).
Secuencia normal	Paso Acción
	1-El usuario accede al menú principal y selecciona la sección "Estadísticas".
	2-El sistema recupera el historial total de escenarios jugados por el usuario.
	3-El sistema calcula los totales de aciertos y fallos en los escenarios
	4-El sistema genera y muestra un gráfico circular con los porcentajes de acierto.
Secuencias alternativas	Paso Acción
	2 ^a - Usuario nuevo: El sistema detecta que el historial está vacío (0 escenarios jugados). Muestra un mensaje amigable: "Aún no tienes estadísticas. ¡Juega tu primer escenario!".
	2b- Error de conexión: No se pueden recuperar los datos del servidor. Se muestra un mensaje de "Intentando reconectar...".

CASO DE USO #2	Visualizar Gráficas de Evolución Temporal
Objetivo en contexto	Ver cómo ha mejorado (o empeorado) el nivel de juego a lo largo del tiempo.
Entradas	Rango de fechas (Último mes, Último año, Personalizado).
Precondiciones	Usuario debe haber iniciado sesión y poseer una Suscripción PRO (o Elite) activa.
Salidas	Gráfica lineal de evolución de ELO o puntuación.
Actores	Usuario Pro o superiores.
Secuencia normal	Paso Acción
	1-El usuario selecciona la pestaña "Evolución" dentro de Estadísticas.
	2-El sistema verifica que el usuario tenga una suscripción activa de nivel PRO o ELITE .
	3-El usuario selecciona un rango de fechas (ej: "Último mes").
	4-El sistema procesa la puntuación (ELO) del usuario día a día en ese periodo.
	5-El sistema renderiza una gráfica lineal interactiva mostrando la tendencia.
Secuencias alternativas	Paso Acción

CASO DE USO #2	Visualizar Gráficas de Evolución Temporal
	2ª-Nivel insuficiente: El usuario tiene nivel Básico. El sistema bloquea el acceso a la gráfica y muestra una ventana emergente ofreciendo mejorar al plan PRO.
	4ª-Sin actividad en el periodo: El usuario selecciona un mes en el que no jugó. El sistema muestra una gráfica plana o vacía con el aviso "Sin actividad en estas fechas".

CASO DE USO #3	Analizar Puntos Débiles por Fase
Objetivo en contexto	Identificar en qué fase del juego (Grande, Chica, Pares, Juego) el usuario suele perder más puntos.
Entradas	Solicitud de análisis.
Precondiciones	Haber jugado escenarios completos que incluyan todas las fases. Usuario debe haber iniciado sesión y poseer una Suscripción PRO (o Elite) activa.
Salidas	Informe desglosado de rendimiento por fase.
Actores	Usuario Pro o superiores.
Secuencia normal	Paso Acción
	1-El usuario solicita el "Análisis táctico".
	2-El sistema verifica que el usuario tenga una suscripción activa de nivel PRO o ELITE .

CASO DE USO #3	Analizar Puntos Débiles por Fase
	3-El sistema examina las rondas perdidas en los últimos X escenarios jugados.
	4-El sistema categoriza las pérdidas según la fase del juego (Grande, Chica, Pares, Juego)
	5-El sistema compara los porcentajes de pérdida contra la media global.
	6-El sistema destaca la fase con peor rendimiento (ej: "Pierdes el 60% de los envites a Grande").
Secuencias alternativas	Paso Acción
	2ª-Nivel insuficiente: El usuario tiene nivel Básico. El sistema bloquea el acceso a la gráfica y muestra una ventana emergente ofreciendo mejorar al plan PRO.
	3ª-Muestra insuficiente: El usuario ha jugado muy pocas partidas para un análisis fiable. El sistema avisa: "Juega al menos 10 partidas para desbloquear el análisis táctico".

CASO DE USO #22	Exportar Histórico a CSV/Excel
Objetivo en contexto	Descargar los datos en crudo para que el usuario pueda hacer sus propios análisis externos.
Entradas	Formato deseado (.csv o .xlsx) y periodo.
Precondiciones	Usuario debe haber iniciado sesión con Suscripción ELITE activa.

CASO DE USO #22	Exportar Histórico a CSV/Excel
Salidas	Archivo descargable con el registro detallado de partidas.
Postcondición si éxito	El archivo se descarga en el dispositivo del usuario.
Actores	Usuario Elite.
Secuencia normal	Paso Acción
	1-El usuario navega a la configuración de datos y pulsa "Exportar Historial".
	2-El sistema valida que el usuario tenga el nivel de suscripción ELITE .
	3-El sistema solicita el formato de salida deseado (.CSV o .XLSX).
	4-El usuario selecciona el formato y confirma.
	5-El sistema compila toda la información de la base de datos en un archivo estructurado.
	6-El sistema inicia la descarga del archivo en el dispositivo del usuario.
Secuencias alternativas	Paso Acción
	2ª- Acceso denegado: El usuario es Básico o Pro. El sistema muestra un mensaje: "La exportación de datos es exclusiva para miembros Elite. Mejora tu plan aquí".

CASO DE USO #22	Exportar Histórico a CSV/Excel
	5ª- Error de generación: El historial es demasiado grande y el proceso tarda demasiado (timeout). El sistema sugiere acotar la exportación por años en lugar de "Todo el historial".

8. Recursos del Proyecto

En esta sección se describen los recursos humanos, materiales, tecnológicos y económicos necesarios para el desarrollo del sistema predictivo de Mus. El objetivo es garantizar que el proyecto disponga de las capacidades y herramientas necesarias para su correcta ejecución conforme a las metodologías de Ingeniería del Software.

8.1. Equipo de trabajo: roles y responsabilidades

El equipo del proyecto está compuesto por diferentes perfiles que asumen roles específicos para asegurar un desarrollo equilibrado y una correcta gestión del ciclo de vida del software. A continuación, se detalla la organización del equipo y las responsabilidades de cada rol.

4.1.1. Jefe de Proyecto (Project Manager)

El Jefe de Proyecto se encarga de:

- Planificar, coordinar y supervisar todas las fases del proyecto.
- Establecer el cronograma de actividades y velar por su cumplimiento.
- Gestionar riesgos, resolver conflictos y asignar recursos.
- Actuar como interlocutor entre el equipo y el profesor/cliente.
- Asegurar la calidad del producto final y la correcta entrega de documentación.

8.1.2. Analista de Requisitos

Responsable de comprender en profundidad las necesidades del usuario y del juego del Mus. Entre sus funciones:

- Recolección y análisis de requisitos funcionales y no funcionales.
- Entrevistas e inspección de reglas del juego y estrategias.
- Elaboración de documentos de especificación, casos de uso y criterios de aceptación.
- Colaboración con el Ingeniero de Pruebas para validar requisitos mediante pruebas.

8.1.3. Arquitecto de Software

Perfil encargado del diseño estructural del sistema:

- Definición de la arquitectura global (como la cliente-servidor).
- Selección de tecnologías (como las bases de datos).
- Diseño de patrones de comunicación (como los controladores).

- Establecimiento de estándares de desarrollo y buenas prácticas.

8.1.4. Desarrolladores

a) Desarrollador tipo 1

- Implementación de la interfaz de usuario.
- Integración con librerías externas.
- Diseño responsivo, accesibilidad y experiencia de usuario.
- Creación de componentes reutilizables y mantenimiento del código.

b) Desarrollador tipo 2

- Implementación de la lógica de negocio.
- Creación y mantenimiento de servicios.
- Gestión de seguridad, validación y control de sesiones.
- Manejo de la base de datos, consultas y modelos.

c) Desarrollador tipo 3

- Investigación de estrategias y probabilidades del Mus.
- Entrenamiento de modelos estadísticos o de desarrollo de IA.
- Simulación de partidas para generar datos adicionales.
- Integración del motor de predicción.
- Ajuste y evaluación del rendimiento del modelo.

8.1.5. Administrador de Base de Datos / Ingeniero de Datos

Encargado de:

- Diseñar el modelo de datos: tablas, relaciones, índices y restricciones.
- Garantizar la integridad de la información.
- Preparar inserts y consultas para análisis y entrenamiento del modelo predictivo.
- Optimizar consultas SQL y gestionar copias de seguridad.

8.1.6. Ingeniero de Pruebas

Responsabilidades:

- Diseñar y ejecutar pruebas unitarias, funcionales, de integración y de sistema.
- Validar el correcto cumplimiento de los requisitos.
- Documentar incidencias y hacer seguimiento de correcciones.
- Elaborar informes de calidad y métricas finales.

8.1.7. Responsable de Documentación y Control de Versiones

Se ocupa de que toda la información generada en el proyecto esté correctamente registrada y organizada:

- Redacción de documentos técnicos, manuales de usuario y anexos.
- Uso de herramientas como Git para controlar versiones del proyecto.
- Organización de entregas de versiones parciales y finales.

8.2. Recursos materiales y tecnológicos

Los recursos del proyecto incluyen tanto hardware como software esencial para el desarrollo, pruebas y despliegue de la aplicación.

8.2.1. Recursos Hardware

- **Equipos de desarrollo:** Ordenadores portátiles o de sobremesa con procesadores de al menos 4 núcleos, 8–16 GB de RAM y almacenamiento SSD.
- **Periféricos:** ratón, teclado, auriculares y otros elementos necesarios para la comunicación y productividad.
- **Servidor de pruebas o despliegue:**
 - Opción en la nube (AWS, Azure, Render, Heroku).

Este hardware permite ejecutar entornos de desarrollo y pruebas del modelo predictivo sin limitaciones técnicas.

8.2.2. Recursos Software

Herramientas de gestión del proyecto

- **GitHub** para planificación y seguimiento ágil (Scrum / Kanban).
- **Microsoft Teams, Google Meet o Discord** para comunicación interna y reuniones.

Entornos de desarrollo

- Visual Studio Code
- PyCharm
- Eclipse
- Oracle
- SQL developer
-

Tecnologías de programación

- **Lo que ve el usuario**
 - Uso de herramientas como Vue, Angular...
- **Lo que sucede por detras (dentro del programa)**
 - Uso de progrmas como Visual Studio, Node.js, Phython...
- **Modelo Predictivo:**
 - NumPy, Pandas
 - Scikit-learn
 - TensorFlow o PyTorch (opcional)

Bases de datos

- SQL Developer
- PostgreSQL
- MySQL

Control de versiones

- Git + GitHub/GitLab
- Repositorio central para control de versiones

Diseño y documentación técnica

- Draw.io, Lucidchart o Figma para diagramas
- Word, LaTeX, Writter o Markdown para documentación
- Swagger / OpenAPI para documentación de las librerías/APIs

Plataformas de despliegue

- Render, Railway, Vercel o Netlify para lo que ve el usuario
- AWS EC2, Heroku o servicios gratuitos para lo que esta dentro del programa

8.3. Presupuesto estimado

Se presenta un presupuesto aproximado basado en valores reales del mercado y proyectos profesionales.

8.3.1. Costes de personal

Se estima que el equipo está formado por 5 miembros, trabajando aproximadamente 100 horas cada uno, valoradas a 15 €/hora (tarifa estándar para trabajos técnicos de formación universitaria).

- **Coste total de personal:**
 $5 \text{ personas} \times 100 \text{ horas} \times 15 \text{ €/hora} = 7.500 \text{ €}$

8.3.2. Costes de software y licencias

La mayoría de herramientas empleadas tienen versiones gratuitas, pero se consideran posibles costes adicionales:

Recurso	Coste estimado
Licencia Figma Pro (3 meses)	36 €
Licencia Jira (plan estándar)	30 €
Herramientas auxiliares	50–100 €
Total estimado licencias: 120–170 €	
(Se utilizará una estimación media de 150 € .)	

8.3.3. Costes de infraestructura

Para el entorno de despliegue y pruebas se contemplan opciones en la nube:

- Servidor virtual básico: 10 €/mes
- Duración estimada del proyecto: 3 meses

Coste estimado infraestructura:
 $10 \text{ €/mes} \times 3 = 30 \text{ €}$

8.3.4. Costes indirectos

Incluyen electricidad, conexión a internet, dispositivos personales, pequeños imprevistos y material fungible.

Coste estimado: 100 €

8.3.5. Presupuesto total

Concepto	Coste
Personal	7.500 €
Software y licencias	150 €
Infraestructura	30 €
Costes indirectos	100 €
Total estimado del proyecto	7.780 €

9. Gestión del Proyecto

9.1. Metodología: Ágil, Scrum

Para el desarrollo de la aplicación de enseñanza y análisis de Mus, hemos seleccionado una metodología Ágil, implementada específicamente a través del marco de trabajo Scrum.

Si bien la planificación temporal global se visualiza mediante un diagrama de Gantt para definir los hitos estratégicos de cara a la inversión (como se detalla en el Plan de Trabajo), la ejecución operativa del desarrollo de software se regirá por ciclos iterativos e incrementales.

Descripción de la aplicación de Scrum en el proyecto:

El proyecto se estructurará en las iteraciones cortas y de duración fija, propias del Scrum denominadas Sprints, tendrán una duración establecida de 2 semanas. Esta frecuencia nos permite liberar incrementos funcionales del software de manera regular, asegurando que el motor de Monte Carlo y la interfaz de usuario evolucionen conjuntamente.

- **Roles:**

Product Owner (Analista de Requisitos): Será responsable de gestionar el *Backlog*, priorizando las historias de usuario según el valor que aporten al aprendizaje del Mus.

Scrum Master (jefe de Proyecto): Facilitará las ceremonias y eliminará impedimentos, asegurando que el equipo cumpla con el cronograma y los hitos de inversión.

Equipo de Desarrollo: El resto de los perfiles técnicos (Arquitecto, Desarrolladores Frontend/Backend e Ingeniero de Pruebas) trabajarán de forma auto-organizada para completar los incrementos.

Nota sobre la Planificación Temporal: Aunque trabajaremos con iteraciones base de 2 semanas, la planificación temporal global (Cronograma Gantt) se ha adaptado a la complejidad variable de los módulos y a la realidad académica del equipo. Esto permite cierta flexibilidad en la duración de las fases de desarrollo sin perder el ritmo de revisión quincenal.

- **Artefactos:** Gestionaremos un Product Backlog (lista priorizada de requisitos y funcionalidades del Mus) y un *Sprint Backlog* (tareas seleccionadas para las 2 semanas actuales).
- **Eventos:** Realizaremos Daily Scrums (reuniones de seguimiento diario de 15 minutos) para sincronizar el trabajo y Sprint Reviews al final de cada ciclo para validar el incremento con los stakeholders (inversores o usuarios de prueba).

Justificación de la elección , por qué Scrum es ideal para este proyecto:

La elección de Scrum responde a la naturaleza técnica y comercial del proyecto de Mus:

1. **Gestión de la Incertidumbre Técnica (Motor Monte Carlo):** El núcleo de nuestra aplicación es un algoritmo complejo de simulación. Es probable que las primeras versiones del motor requieran ajustes de rendimiento o precisión en las predicciones. Una metodología en cascada sería demasiado rígida y arriesgada, ya que un error en el diseño del algoritmo podría no detectarse hasta el final. Scrum nos permite iterar sobre el motor de recomendación: primero lanzamos una versión básica, la probamos, y en el siguiente Sprint refinamos la heurística basándonos en resultados reales. Esto se alinea con el principio de que la planificación no es estática, sino que evoluciona.
2. **Adaptabilidad a los Requisitos del Usuario:** En una herramienta educativa, la "usabilidad" es un requisito no funcional crítico. Los usuarios deben sentirse cómodos introduciendo sus cartas manualmente. Scrum permite desarrollar prototipos funcionales de la interfaz de entrada de datos y recibir feedback inmediato. Si el sistema de entrada manual resulta tedioso, podemos pivotar y mejorarlo en el siguiente Sprint, evitando desarrollar un producto final que el usuario no quiera utilizar.
3. **Entrega de Valor Temprana (Time-to-Market):** Para una ronda de inversión, es vital demostrar progreso tangible. Scrum nos permite tener un Producto Mínimo Viable (MVP) muy pronto. Por ejemplo, podríamos finalizar primero el módulo de "Cálculo de probabilidades" antes de tener listo el "Modo Entrenamiento". Esto demuestra capacidad de ejecución y reduce el riesgo percibido por los inversores, ya que se entregan productos de forma continua.
4. **Control de Riesgos:** Al dividir el proyecto en bloques manejables, reducimos el impacto de los errores. Si una funcionalidad específica del juego (ej. la gestión de los "pares") se retrasa, no paraliza todo el proyecto, sino que se prioriza en el siguiente Sprint sin afectar a la arquitectura global.

9.2. Gestión de Riesgos

En el desarrollo de software, "no hay proyectos sin riesgo, solo proyectos sin gestión del riesgo". Para asegurar la viabilidad de nuestra inversión y la entrega exitosa de la aplicación de aprendizaje de Mus, hemos adoptado una estrategia proactiva. A diferencia de una gestión reactiva, que actúa cuando el problema ya existe, nuestro enfoque se centra en la prevención: identificar incertidumbres antes de que se conviertan en crisis, evaluando su probabilidad e impacto para minimizar su efecto en el proyecto.

Nuestro enfoque se basa en el principio de Pareto: asumimos que el 80% del riesgo real del proyecto provendrá del 20% de las amenazas identificadas, por lo que focalizaremos nuestros recursos en los riesgos críticos descritos a continuación.

9.2.1. Planificación Inicial y Estrategias de Respuesta

El proceso de planificación inicial ha consistido en identificar amenazas en tres dimensiones clave: **Riesgos del Proyecto** (presupuesto, plazos), **Riesgos Técnicos** (calidad, dificultad de implementación) y **Riesgos del Negocio** (viabilidad comercial).

Para cada riesgo identificado, hemos seleccionado una de las cuatro estrategias de respuesta estándar:

1. **Evitar:** Cambiar el plan para eliminar la amenaza.
2. **Reducir (Mitigar):** Acciones proactivas para bajar la probabilidad o el impacto.
3. **Asumir (Aceptar):** Entender que el riesgo puede ocurrir y tener un plan de contingencia listo.
4. **Transferir:** Desplazar la responsabilidad a terceros (ej. proveedores cloud) .

9.2.2. Matriz de Riesgos y Plan de Acción (RSGR)

A continuación, se presenta la **Tabla de Riesgos Priorizada**. Este documento vivo constituye el núcleo de nuestro *Plan de Reducción, Supervisión y Gestión del Riesgo (RSGR)*. Los riesgos se han ordenado según su **Exposición** (Probabilidad x Impacto), definiendo umbrales de severidad (Catastrófico, Crítico, Serio, Menor)

R06	Scope Creep (Alcance no controlado): Tentación de añadir funcionalidades complejas (como multijugador u opciones de accesibilidad) desviando el objetivo didáctico y el foco de las cosas fundamentales para la primera versión, centrándose en su lugar en cosas que deberían plantearse en el futuro.	Proyecto	Media	Serio	Evitar	Mitigación: Congelar requisitos tras el análisis. Mantener un <i>Backlog</i> estricto. Contingencia: Mover cualquier nueva idea a una lista de "Versión 2.0" futura.
R07	Incomprensión de recomendaciones: El usuario recibe la "mejor jugada" pero no entiende <i>por qué</i> , fallando el objetivo didáctico.	Negocio	Media	Medio	Reducir	Mitigación: Incluir funcionalidad de "Explicación" simple (ej. probabilidades de éxito). Contingencia: Añadir ayudas visuales estáticas si la explicación dinámica es muy compleja.
R08	Problemas de Integración (Gráficos): Dificultad técnica para visualizar estadísticas y tanteo con librerías externas.	Técnico	Baja	Menor	Asumir	Mitigación: Utilizar librerías estándar bien documentadas. Contingencia: Sustituir gráficos dinámicos por tablas numéricas simples.
						Mitigación: Ajustar la heurística

ID	Riesgo Identificado	Tipo	Prob.	Impacto	Estrategia	Plan de Acción
R01	Brecha de habilidades en IA: El equipo carece de experiencia profunda en algoritmos Monte Carlo, lo que puede bloquear el desarrollo del núcleo.	Proyecto	Alta	Crítico	Reducir	Mitigación: Realizar un "Spike" (investigación técnica) en el Sprint 0. Programación en pareja (Pair Programming) para las partes complejas. Contingencia: Simplificar el motor a un árbol de decisión básico si el Monte Carlo no es viable en fecha.
R02	Incoherencia en reglas del Mus: Codificar incorrectamente la lógica (ej. prioridades de pares, mano vs poste) invalidando las recomendaciones.	Técnico	Media	Crítico	Reducir	Mitigación: Definición exhaustiva de casos de prueba basados en el reglamento oficial antes de codificar. Contingencia: Auditoría de código cruzada centrada exclusivamente en la lógica de negocio.
R03	Fricción en entrada manual: El usuario abandona la aplicación porque introducir las cartas manualmente resulta lento o tedioso.	Negocio	Alta	Serio	Evitar	Mitigación: Diseño de interfaz con gestos rápidos y selectores predictivos. Prototipado en Sprint 1 para validación. Contingencia: Implementar escenarios predefinidos para reducir la necesidad de entrada manual constante.
R04	Sobrecarga de computación: El motor Monte Carlo consume demasiados recursos del móvil, provocando lentitud, cierres y sobrecalentamiento	Técnico	Media	Serio	Reducir	Mitigación: Optimizar el tiempo de cálculo o número de simulaciones dinámicamente según la potencia del móvil. Contingencia: Mover el cálculo a un servidor en la nube si el rendimiento local es

R13	Consumo de Batería: El motor de cálculo intensivo drena la batería del usuario rápidamente, provocando desinstalaciones.	Técnico	Media	Medio	Asumir	Mitigación: Optimizar el código y pausar el motor cuando la app está en segundo plano. Contingencia: Aviso al usuario: "El análisis profundo consume batería" si activa el modo de máxima precisión.
R14	Desbalanceo de la dificultad: Los escenarios de entrenamiento "Fáciles" son demasiado difíciles o los "Difíciles" son imposibles, desmotivando al alumno.	Negocio	Media	Serio	Reducir	Mitigación: Beta testing con usuarios reales de distintos niveles (novatos vs expertos) para calibrar la IA. Contingencia: Ajustar los parámetros de dificultad mediante un parche rápido tras recibir feedback.
R15	Infracción de Copyright (Baraja): Utilizar los diseños icónicos de las cartas (ej. Heraclio Fournier) Se suele pensar que son de dominio público sin embargo su uso sin licencia, provocaría el bloqueo de la App en las tiendas.	Negocio	Baja	Catastrófico	Evitar	Mitigación: Diseñar un set de cartas propio "estilo español" pero original o usar assets de dominio público (Creative Commons). Contingencia: Reemplazar los gráficos de las cartas por versiones simplificadas (números y símbolos vectoriales) de emergencia.
R16	Privacidad de datos : Si guardamos estadísticas de uso o perfiles de jugador, podemos incumplir la normativa de protección de datos si no se gestiona bien el consentimiento, o cometemos errores en la gestión de la privacidad por falta de conocimiento legal.	Negocio	Media	Serio	Evitar	Mitigación: Diseñar la app para que todos los datos se guarden solo en local (en el dispositivo del usuario) y no en servidores externos. Contingencia: Implementar un banner de cookies y política de privacidad genérica si decidimos subir datos a la nube. Recibir asesoramiento legal con respecto a la privacidad
R17	Curva de aprendizaje de la App: El usuario sabe jugar al Mus pero no entiende cómo usar la herramienta de análisis (la herramienta es más compleja que el juego).	Negocio	Media	Alto	Reducir	Mitigación: Incluir un tutorial guiado la primera vez que se abre la app. Contingencia: Simplificar la interfaz ocultando las estadísticas avanzadas por defecto (modo "Experto" oculto). Testear el comportamiento de los usuarios con respecto a la interfaz
R12	(persistencia local) se corrompe tras una actualización de la app, frustrando al usuario.	Técnico	Baja	Medio	Transferir	Contingencia: Implementar una función de "Reset de estadísticas" para recuperar la funcionalidad de la app si los datos se corrompen.

9.2.3. Proceso de Seguimiento y Control

La gestión de riesgos no termina con la creación de la tabla anterior. Para asegurar que el plan se mantiene eficaz y actualizado, implementaremos un proceso de Monitorización Continúa integrado en nuestra metodología Scrum:

1. **Revisión en Sprint Retrospectives:** Al finalizar cada ciclo de 2 semanas, dedicaremos un espacio para revisar la lista de riesgos ("Risk Scrubbing"). Evaluaremos:
 - ¿Se ha materializado algún riesgo?

- ¿Son eficaces las acciones de mitigación implementadas (ej. ¿ha bajado la latencia del Monte Carlo?)?
 - ¿Han surgido nuevos riesgos derivados de los cambios recientes?
2. **Métricas e Indicadores:** Utilizaremos métricas de calidad para vigilar los riesgos técnicos (ej. tiempo medio de respuesta del algoritmo) y métricas de proyecto para los riesgos de planificación (ej. velocidad del equipo). Esto nos permite disparar las alarmas antes de que el impacto sea irreversible.
 3. **Actualización del Plan:** El documento RSGR será actualizado mensualmente. Un riesgo que ha sido mitigado bajará de prioridad, mientras que nuevos riesgos emergentes (ej. cambios en las políticas de la App Store) serán añadidos y evaluados.

9.2.4. Conclusión sobre la Gestión del Riesgo

Este plan de riesgos demuestra a los inversores que el equipo tiene una visión optimista del producto, pero también realista y definida. Entendemos que la complejidad del motor Monte Carlo y la usabilidad de la entrada manual son nuestros mayores desafíos.

Al tener identificados estos puntos críticos y sus correspondientes planes de contingencia, transformamos la incertidumbre en variables gestionables. Un buen plan de riesgos no elimina los problemas, pero evita las crisis, asegurando una toma de decisiones informada y ágil.

9.3. Control de Calidad (SQA)

La calidad no se "añade" al final del proyecto, sino que se construye durante todo el proceso. Para asegurar que nuestra aplicación de Mus es robusta y fiable, implementaremos un plan de Garantía de Calidad del Software (SQA) integrado en cada Sprint, combinando revisiones preventivas y pruebas exhaustivas, que desarrollaremos en los siguientes puntos.

9.3.1. Revisiones Técnicas Formales (RTF)

Realizaremos las siguientes revisiones sistemáticas para detectar errores antes de que lleguen al código final.

- **Revisiones de Diseño:** Antes de implementar una nueva funcionalidad compleja (ej. la lógica de los "pares"), un segundo desarrollador revisará el diseño propuesto. Esto asegura que la lógica del juego sea correcta antes de escribir código.
- **Revisiones de Código :** Ningún código se integrará en la rama principal sin haber sido revisado y aprobado por otro miembro del equipo mediante "Pull Requests". Esto fomenta la propiedad colectiva del código y reduce el riesgo de introducir bugs críticos.

9.3.2. Estrategia de Pruebas (Testing)

Dada la complejidad matemática de nuestro motor de recomendación, la estrategia de pruebas es vital. Aplicaremos diferentes niveles de testeo:

- **Pruebas Unitarias:** Nos centraremos en validar los componentes más pequeños y críticos, especialmente las reglas del juego. Ejemplo: Verificar que la función `es_31_real()` (función para determinar si la mano es juego o no) devuelve `true` solo cuando corresponde.
- **Pruebas de Sistema:** Validaremos el flujo completo de una partida, asegurando que la entrada manual de cartas actualiza correctamente el estado del juego y el motor de recomendación.
- **Pruebas de Aceptación (Validación con Usuarios):** En cada Sprint Review, probaremos la aplicación con usuarios reales para validar que las recomendaciones de la IA son útiles y que la interfaz es cómoda.

9.3.3. Estándares y Métricas

Para mantener la coherencia técnica, hemos definido:

- **Estándares de Código:** Utilizaremos una guía de estilo común (ej. PEP8 si usamos Python, o la estándar de Java/Kotlin) y herramientas de análisis estático para asegurar que todo el código sea legible y mantenible.
- **Métricas de Calidad:** Monitorizaremos indicadores clave como la Densidad de Defectos (errores encontrados por Sprint) para identificar módulos problemáticos y la **Cobertura de Código** para asegurar que las reglas críticas del Mus están testeadas.

9.4. Gestión de Cambios

En un proyecto de desarrollo de software, el cambio es inevitable. Sin embargo, un cambio no controlado conduce al caos y al retraso. Para evitar esto, hemos establecido un Procedimiento de Control de Cambios formal pero ágil, integrado en nuestra gestión del *Backlog*.

Nuestro procedimiento para manejar modificaciones en el alcance o los requisitos consta de cuatro pasos:

1. **Petición de Cambio (RFC):** Cualquier propuesta de cambio (venga de un inversor, un usuario o un miembro del equipo) debe formalizarse como una "Historia de Usuario" o "Tarea" en nuestro tablero de gestión (Trello/Jira). No se aceptan cambios verbales.
2. **Análisis de Impacto:** Antes de aceptar el cambio, el equipo técnico evalúa su coste:
 - ¿Afecta a la arquitectura del motor Monte Carlo?
 - ¿Requiere tirar código ya hecho?
 - ¿Cuánto esfuerzo (puntos de historia) supone?
3. **Aprobación y Priorización:**
 - Si el cambio es crítico (ej. un error en las reglas), se incorpora al Sprint actual si es posible, o al siguiente con máxima prioridad.
 - Si es una mejora (ej. añadir animaciones), se envía al Product Backlog y se prioriza frente al resto de tareas pendientes.

- Si es fuera de alcance (ej. multijugador), se descarta o se mueve a una lista de deseos para futuras versiones.
4. **Implementación y Trazabilidad:** Una vez aprobado, el cambio sigue el flujo de desarrollo normal. Utilizamos un sistema de Control de Versiones (Git) que nos permite mantener la trazabilidad: cada cambio en el código está asociado a una tarea específica, asegurando que sabemos quién cambió *qué* y por qué .

9.5. Plan de Comunicación y Herramientas

La comunicación efectiva es el "pegamento" que mantiene unido al proyecto. Para evitar silos de información y malentendidos (especialmente críticos en un equipo donde se integran componentes complejos como el motor de IA y la interfaz de usuario) hemos establecido un Plan de Comunicación Híbrido que prioriza la transparencia y la rapidez.

9.5.1. Estrategia de Reuniones (Rituales Scrum)

Para garantizar la sincronización sin caer en el exceso de reuniones improductivas, nos ceñiremos estrictamente a los eventos del marco Scrum:

- **Daily Stand-up (Virtual):** Una actualización rápida de 15 minutos cada mañana vía chat o videoconferencia. Cada miembro responde tres preguntas: ¿Qué hice ayer? ¿Qué haré hoy? ¿Tengo algún bloqueo? Esto permite detectar riesgos (como el R01 o R05 mencionados anteriormente) en menos de 24 horas.
- **Sprint Planning (Inicio de iteración):** Reunión presencial quincenal para definir el objetivo del Sprint y repartir las tareas del Backlog.
- **Sprint Review & Retrospective (Fin de iteración):** Sesión clave para mostrar el avance real (demo de la app) y analizar qué procesos internos debemos mejorar para el siguiente ciclo.

9.5.2. Herramientas de Gestión y Colaboración

Hemos seleccionado un stack tecnológico que nos permite centralizar la información y mantener la trazabilidad del trabajo:

- **Gestión de Tareas (GitHub Projects):** Utilizaremos los tableros Kanban integrados en GitHub para visualizar el flujo de trabajo. Cada tarea (ej. "Implementar cálculo de pares") pasará por los estados: To Do, In Progress, In Review, Done. Al vincular las tareas directamente a las ramas de código (Issues y Pull Requests), automatizamos el seguimiento del progreso.
- **Comunicación Asíncrona (Discord):** Para la comunicación diaria, utilizaremos canales temáticos (#dev-backend, #diseño-ui, #general). Esto reduce el ruido del correo electrónico y permite que las discusiones técnicas queden documentadas y sean fáciles de buscar.
- **Gestión Documental (GitHub Wiki / Repositorio):** Toda la documentación del proyecto, incluyendo este plan, el SRS y las actas de reuniones, se almacenará en el repositorio compartido, asegurando el control de versiones y que todo el equipo trabaja sobre la última versión aprobada.

9.5.3. Reportes y Seguimiento

Para mantener informados a los inversores y supervisores sobre la salud del proyecto, generaremos los siguientes reportes:

- **Gráfico de Burndown:** Un indicador visual que muestra la velocidad de trabajo del equipo y si llegaremos a tiempo a la entrega final. Es nuestra herramienta principal de transparencia.
- **Informe de Estado del Sprint:** Un resumen quincenal que incluye:
 - Funcionalidades completadas (Entregables).
 - Estado de los Riesgos Críticos (del plan RSGR).
 - Desviaciones en el cronograma y acciones correctoras tomadas.

Plan de Pruebas

1. Introducción

El presente *Plan de Pruebas* define la estrategia, alcance, responsabilidades, actividades, herramientas, criterios de aceptación y metodología de validación del **Sistema Predictivo de Mus**, cuyo principal objetivo es analizar una mano de cartas y generar una recomendación óptima de jugada basándose en un modelo predictivo entrenado con datos históricos de partidas.

El propósito de este plan es garantizar que el sistema cumpla con los requisitos funcionales y no funcionales establecidos, asegurando la fiabilidad, robustez, mantenibilidad y precisión del software.

2. Tipos de Pruebas

Este plan adopta un enfoque incremental y sistemático, estructurado en las siguientes categorías fundamentales:

2.1 Pruebas Unitarias

Objetivo

Verificar el correcto funcionamiento individual de cada módulo, función o método del sistema de forma aislada.

Elementos a evaluar

- **Preprocesamiento de datos**
 - Funciones de limpieza y transformación de datos (handling de NaN, encoding, normalización, detección de valores atípicos).
 - Verificación del correcto parsing de cartas y jugadas.
- **Cálculo de características (“features”)**
 - Generación de indicadores estadísticos de manos de Mus.
 - Funciones de evaluación de probabilidades previas (probabilidad de tener pares, juego, etc.).
- **Modelo predictivo**
 - Cálculo de predicciones.
 - Carga correcta del modelo entrenado.
 - Comprobación de que el modelo produce outputs dentro del dominio permitido.

- **Motor de reglas del Mus**
 - Comprobación de reglas condicionadas (grande, chica, pares, juego).
 - Consistencia de resultados lógicos.
- **API / Backend**
 - Endpoints independientes.
 - Validación de formatos de entrada y salida.

Criterios de éxito

- El 100% de los tests unitarios deben ejecutarse sin errores.
- Cobertura mínima recomendada: $\geq 80\%$ del código crítico.

2.2 Pruebas de Integración

Objetivo

Verificar que los distintos módulos interactúan correctamente y que la información fluye sin inconsistencias.

Elementos a integrar y probar

- **Interacción entre el preprocesador y el modelo predictivo.**
 - Confirmar que los datos se transforman correctamente antes de llegar al modelo.
- **Comunicación entre base de datos y sistema de análisis.**
 - Verificación de consultas, tiempo de respuesta y coherencia de datos recibidos.
- **Integración entre backend y frontend.**
 - Comprobación del envío y recepción correcta de predicciones.
 - Validación del formato JSON en respuestas.
- **Integración entre motor de reglas y predicciones ML.**
 - Verificar que no se generan jugadas inconsistentes según las reglas del Mus.
- **Pipeline completo de predicción**
 - Entrada del usuario →
 - Validación →
 - Preprocesamiento →
 - Cálculo del modelo →

- Aplicación de reglas →
- Entrega de la jugada recomendada.

Criterios de éxito

- Todos los módulos deben ser capaces de comunicarse sin fallos.
- El pipeline completo debe ejecutarse sin interrupciones.
- No debe haber pérdida ni alteración no prevista de datos.

2.3 Pruebas del Sistema

Objetivo

Validar el comportamiento del sistema como un todo bajo condiciones reales de operación.

Aspectos a evaluar

Requisitos funcionales

- El sistema debe aceptar una mano válida de Mus.
- El sistema debe generar una predicción para la jugada recomendada.
- En caso de datos inválidos, el sistema debe devolver un mensaje de error claro.
- El sistema debe permitir revisar estadísticas del modelo.

Requisitos no funcionales

- **Rendimiento:**
 - Tiempo máximo de respuesta: < **500 ms** por predicción.
- **Usabilidad:**
 - Interfaz clara y coherente con las reglas del juego.
 - Mensajes comprensibles para el usuario.
- **Robustez:**
 - El sistema no debe fallar ante entradas mal formadas.
- **Escalabilidad:**
 - Capacidad para procesar varias peticiones simultáneas.
- **Disponibilidad:**
 - El sistema debe funcionar de forma consistente durante períodos prolongados.

Escenarios principales

- Usuario introduce una mano perfectamente válida → predicción correcta.
- Usuario introduce una mano incompleta o inválida → error bien manejado.
- Sobrecarga de peticiones simultáneas → el sistema mantiene rendimiento estable.
- Conexión temporal perdida hacia la base de datos → recuperación adecuada.

2.4 Pruebas de Aceptación

Objetivo

Validar, desde el punto de vista del cliente/usuario final, que el sistema cumple los requisitos y expectativas del proyecto.

Criterios de aceptación

- La precisión del modelo debe superar el umbral definido (ej.: $\geq 80\%$).
- Las recomendaciones deben ser coherentes con las reglas reales del Mus.
- La interfaz debe permitir al usuario obtener una predicción en ≤ 3 clics.
- La experiencia general debe considerarse satisfactoria por parte de los evaluadores.

Procedimiento

- Sesiones guiadas con usuarios (profesores o compañeros).
- Evaluación mediante checklist basada en requisitos.
- Registro de incidencias y retroalimentación.

3. Estrategia de Validación

La validación se basa en los siguientes pilares:

3.1 Validación del Modelo Predictivo

Técnicas a aplicar

- **Validación cruzada (k-fold)**
- **Conjunto de test independiente**
- **Métricas del modelo según el tipo de predicción:**
 - *Accuracy*

- *F1-score*
- *ROC-AUC* (si aplica)
- *Mean Absolute Error* (si predice valores numéricos)
- **Detección de sobreajuste (overfitting)**

Criterios de validación del modelo

- El modelo debe presentar estabilidad entre entrenamiento y test (variación \leq 10%).
- No debe mostrar sesgos hacia manos o patrones concretos.
- Debe responder de forma consistente a datos no vistos.

3.2 Validación de Requisitos Funcionales

Cada requisito del documento de especificación debe estar asociado a uno o más casos de prueba.

Ejemplo:

Requisito	Caso de prueba	Criterio de aceptación
RF-01: El sistema debe generar la jugada recomendada.	CP-001: Predicción con mano válida.	Output dentro del dominio permitido.
RF-02: Validar la entrada del usuario.	CP-014: Mano inválida.	Error mostrado adecuadamente.

3.3 Validación de Requisitos No Funcionales

- **Rendimiento:** mediciones de latencia y tiempos por request.
- **Usabilidad:** pruebas con usuarios reales, análisis de intuición de la interfaz.
- **Seguridad:** validación de sanitización de entrada para evitar inyecciones (según corresponda).
- **Mantenibilidad:** revisión de estructura del código, modularización, documentación.

4. Herramientas de Testing

A continuación se presenta un inventario detallado de herramientas recomendadas según la naturaleza del sistema:

4.1 Para pruebas unitarias

- **pytest** (Python): ejecución de pruebas ágiles, fixtures, mocks.

- **unittest** (Python): framework estándar para pruebas formales.
- **pytest-cov**: informe de cobertura del código.
- **pandas-testing**: comparación estructural de DataFrames.
- **mock / monkeypatch**: simulación de dependencias externas.

4.2 Para pruebas de integración

- **Postman / Insomnia**: validación de APIs REST y flujos de peticiones.
- **Docker Compose**: simulación de entornos completos (BD + backend + modelo).
- **pytest con plugins de API testing**.

4.3 Para pruebas de sistema

- **Selenium WebDriver**: pruebas automáticas del frontend.
- **Locust**: pruebas de rendimiento bajo carga.
- **JMeter**: stress test y carga sostenida.

4.4 Para pruebas del modelo

- **scikit-learn (metrics)**: métricas para valores y clasificaciones.
- **MLflow**: trazabilidad de experimentos.
- **TensorBoard**: monitorización si se usan redes neuronales.

4.5 Integración Continua

- **GitHub Actions**: correr tests automáticamente en cada commit.
- **GitLab CI/CD**: pipelines para testing, build y despliegue.
- **Jenkins**: servidores de integración continua auto-gestionados.

CI/CD permite:

- Detectar fallos antes de la entrega.
- Automatizar pruebas unitarias e integración.
- Asegurar calidad constante del sistema.

5. Criterios de Éxito del Plan de Pruebas

Para considerar que el sistema está listo para entrega:

5.1 A nivel técnico

- $\geq 80\%$ de cobertura de código.
- 0 errores críticos o bloqueantes.
- Precisión del modelo \geq umbral establecido.
- Tiempos de respuesta < 500 ms en promedio.

5.2 A nivel funcional

- El sistema predice correctamente jugadas coherentes.
- Todas las reglas del Mus se respetan.
- Todos los flujos de usuario funcionan de principio a fin.

5.3 A nivel de usuario

- Los evaluadores pueden completar la tarea sin ayuda.
- No se registran errores graves durante las pruebas.
- La interfaz es entendible, rápida y clara.

Perfecto. A continuación te entrego un paquete completo y profesional con:

1. **Casos de prueba detallados (más de 25)**
2. **Matriz de trazabilidad Requisitos \leftrightarrow Pruebas**
3. **Diagrama del flujo de validación del modelo**
4. **Ejemplos reales de tests unitarios en Python (pytest)**
5. **Ejemplo de pruebas de integración (API)**