LUDOPATIA UCM

Autores: Xuan Ye, Unai Peral, Pelayo Pérez, Alfonso Rubio, Alejandro Dobarro, Tomás Vázquez, Mario de la Torre y Ximena Pizarro.

Indice

1. Introducción

1.1. Propósito

Este documento especifica todos los aspectos técnicos y de diseño del proyecto **LUDOPATÍA.** Está dirigido a los desarrolladores y clientes, con el objetivo de guiar el avance del proyecto y presentar la funcionalidad y el aspecto final que tendrá el producto.

1.2. Alcance

El proyecto consiste en una plataforma para adolescentes y jóvenes que reproduce las mecánicas de un casino con el fin de **prevenir la ludopatía.** La plataforma implementa de forma sencilla información sobre probabilidades, valor esperado y riesgos al realizar apuestas.

Los usuarios que dispongan de una cuenta podrán usar una **moneda virtual** sin valor real para realizar simulaciones de distintos juegos. Cada usuario tendrá una herramienta para el seguimiento del balance y recursos dedicados a la concienciación sobre el juego responsable.

Finalmente, cada cierto tiempo, aparecerán **mensajes personalizados** en forma de pop-up que se cerrarán automáticamente después de 5 segundos.

2. Antecedentes y justificación

El número de jóvenes entre 18 y 25 años que apuestan **aumenta** anualmente debido a la accesibilidad de los casinos, la normalización de las apuestas deportivas y la falta de educación financiera.

Los datos muestran que esto solo va a ir a peor, por lo que el desarrollo de esta plataforma dedicada a **frenarlo,** con recursos personalizados, no solo es necesario sino que es urgente.

3. Objetivos generales y específicos

Desarrollar una plataforma interactiva que prevenga y reduzca la participación de jóvenes en apuestas, mediante el **seguimiento personalizado** de la actividad en dicha plataforma.

Implementar contenidos educativos para aumentar el conocimiento real sobre el funcionamiento de las apuestas.

Identificar comportamientos de riesgo en los usuarios registrados que estén usando la moneda virtual mediante el seguimiento del balance y la actividad del mismo.

4. Alcance del Proyecto

Este apartado define los límites del sistema **LUDOPATÍA**, detallando qué funcionalidades se desarrollarán para cumplir con el objetivo de prevenir la adicción al juego en jóvenes y qué aspectos quedarán fuera de esta iteración.

4.1. Funcionalidades que se incluirán

El sistema se centrará en la simulación, educación y monitorización del usuario. Las funcionalidades clave son:

- **Gestión de Usuarios y Cuentas:**
    - Sistema de registro e inicio de sesión para permitir el seguimiento individualizado.
- Asignación de una cartera con **moneda virtual sin valor real** para cada usuario registrado.
- **Simulador de Juegos de Azar:**

- Implementación de mecánicas de distintos juegos de casino (ej. ruleta, tragaperras) para realizar simulaciones.
- Visualización transparente de las **probabilidades reales** de ganar/perder y el **Valor Esperado (EV)** en cada apuesta realizada.

- **Herramienta de Seguimiento de Balance:**

- Panel de control (Dashboard) donde el usuario puede ver la evolución de su saldo virtual e historial de apuestas.
- Identificación automática de comportamientos de riesgo basada en la actividad del balance.

- **Módulo de Concienciación y Educación:**

- Recursos educativos integrados sobre el funcionamiento matemático de las apuestas y educación financiera.

- **Sistema de Intervención Automatizada (Pop-ups):**
    - Generación de mensajes emergentes personalizados basados en el tiempo de juego o comportamiento.

- Temporizador automático para el cierre de estos mensajes tras 5 segundos.

4.2. Límites del sistema

Es crucial definir qué **no** hará el sistema para gestionar las expectativas del cliente y los desarrolladores:

- **Sin transacciones monetarias reales:** Bajo ninguna circunstancia se implementarán pasarelas de pago, depósitos bancarios ni retiros de dinero real. La moneda es estrictamente virtual.
- **No es una herramienta clínica:** La plataforma es preventiva y educativa. No sustituye a una terapia psicológica profesional ni diagnostica clínicamente la ludopatía, solo identifica patrones de riesgo.

- **Juegos Multijugador en tiempo real:** El alcance inicial se limita a la interacción Usuario-Sistema para la simulación; no se incluirán partidas de póker u otros juegos contra otros usuarios en tiempo real (a menos que se especifique una ampliación futura).
- **Aplicación Nativa Móvil:** El desarrollo se centrará en una plataforma web accesible desde navegadores, no en apps nativas (iOS/Android) en esta primera fase.

4.3. Entregables principales

Al finalizar el ciclo de desarrollo, se entregarán los siguientes artefactos:

1. **Código Fuente:** Repositorio completo de la plataforma web (Frontend y Backend).
2. **Base de Datos:** Scripts de creación y población inicial de la base de datos para usuarios y registros de actividad.
3. **Documentación Técnica:**
    - Documento de Especificación de Requisitos Software (SRS).
    - Diagramas de diseño (UML: Clases, Casos de Uso, Secuencia).
4. **Manual de Usuario:** Guía explicativa sobre cómo interpretar las estadísticas de probabilidad y uso del simulador.
5. **Informe de Pruebas:** Resultados de las pruebas funcionales, especialmente la validación de los cálculos matemáticos de probabilidad y el correcto funcionamiento de los pop-ups temporizados.

4.4. Criterios de aceptación

El producto se considerará aceptado si cumple con las siguientes condiciones:

- **Precisión Matemática:** Los cálculos de probabilidad y valor esperado mostrados en la simulación deben ser matemáticamente correctos y verificables.
- **Comportamiento de la Interfaz:** Los mensajes de concienciación (pop-ups) deben aparecer según la lógica programada y cerrarse automáticamente a los 5 segundos exactos, sin intervención del usuario si así se requiere.
- **Seguridad de Datos:** La información de las cuentas de usuario y sus patrones de comportamiento debe almacenarse de forma segura, respetando la privacidad de los jóvenes.
- **Usabilidad:** La interfaz debe ser intuitiva y atractiva para el rango de edad objetivo (18-25 años), facilitando la comprensión de los datos financieros complejos.
- **Integridad de la Moneda Virtual:** El sistema debe impedir que el balance sea "infinito" sin reinicios controlados, para simular fielmente la pérdida de recursos (la bancarrota virtual).

5. Requerimientos del sistema:

5.1 Requisitos funcionales

Los requisitos funcionales son aquellos que describen qué es lo que tiene que hacer el software. Definen las funciones que el sistema debe tener.

Utilizando la metodología MoSCoW:

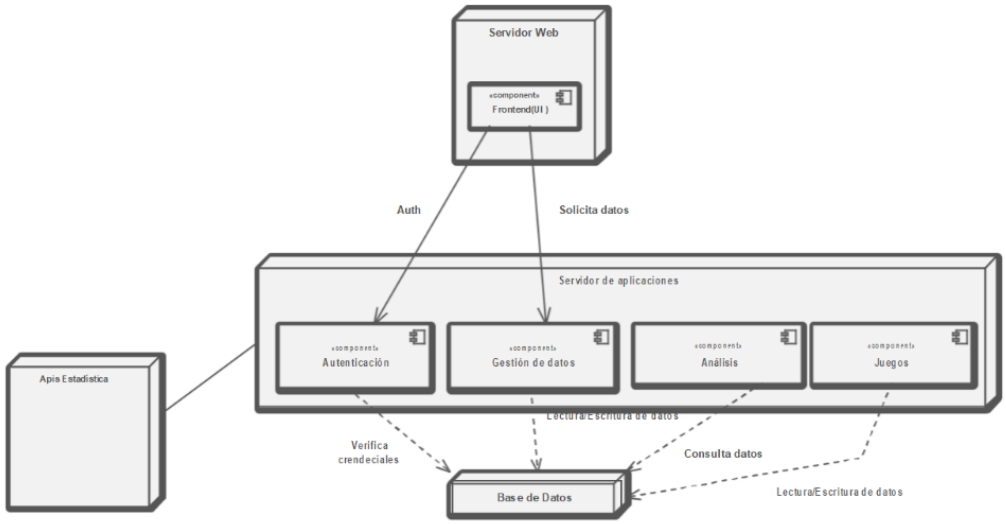
- ➢ M: Must have (obligatorio)
- ➢ S: Should have (debería tener)
- ➢ C: Could have (no esenciales, pero sería óptimo)
- ➢ W: Will not have (Funcionalidades excluidas temporalmente)

ID	Requisito Funcional	Descripción	Prioridad (MoSCoW)
RF01	Registro de usuarios	Los usuarios se registran proporcionando nombre, correo electrónico y contraseña.	M
RF02	Inicio de Sesión	Los usuarios registrados pueden iniciar sesión mediante correo y contraseña.	M

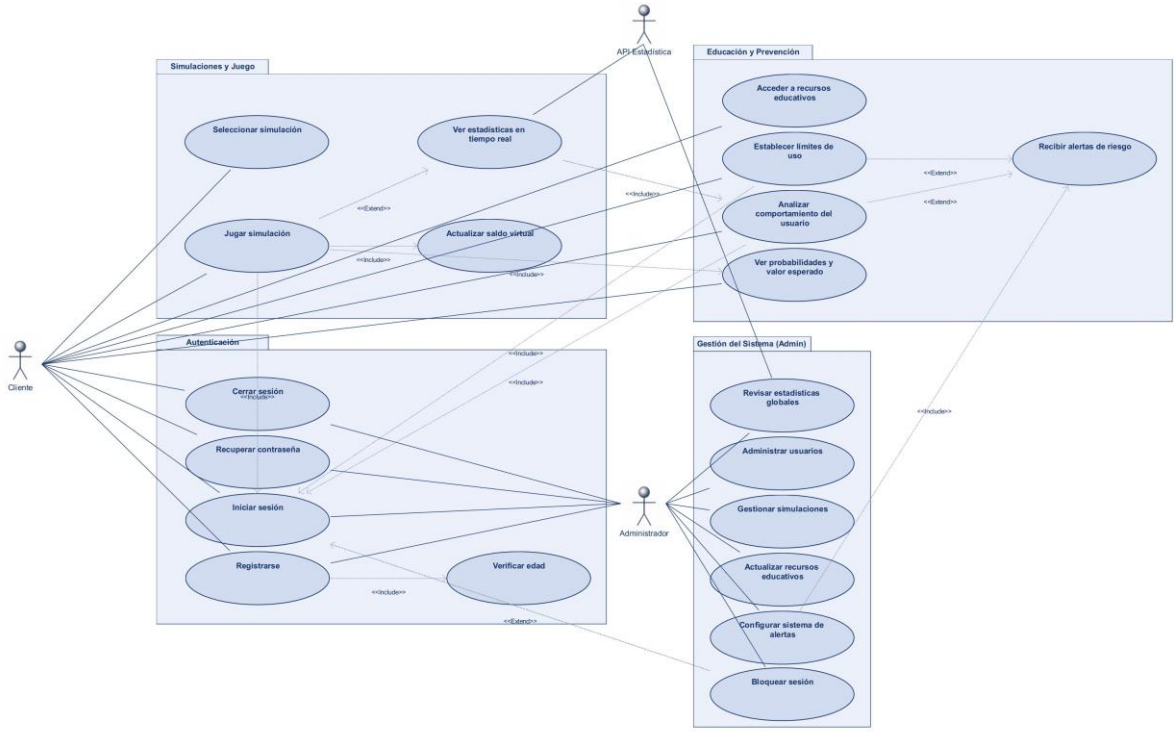
RF03	Verificación de edad	Se deberá verificar que el usuario declara ser mayor de edad antes de acceder a las simulaciones.	M
RF04	Visualización de probabilidades y valor esperado	Se mostrará, antes de iniciar cada simulación, la probabilidad de ganar, y el valor esperado de cada juego.	M
RF05	Simulación de mecánicas del juego	El sistema permitirá al usuario participar en simulaciones de juegos típicos (p. ej., ruleta, tragaperras, blackjack) utilizando monedas virtuales sin valor económico real.	M

6. Arquitectura y diseño del sistema:

6.1 Diagrama general de arquitectura



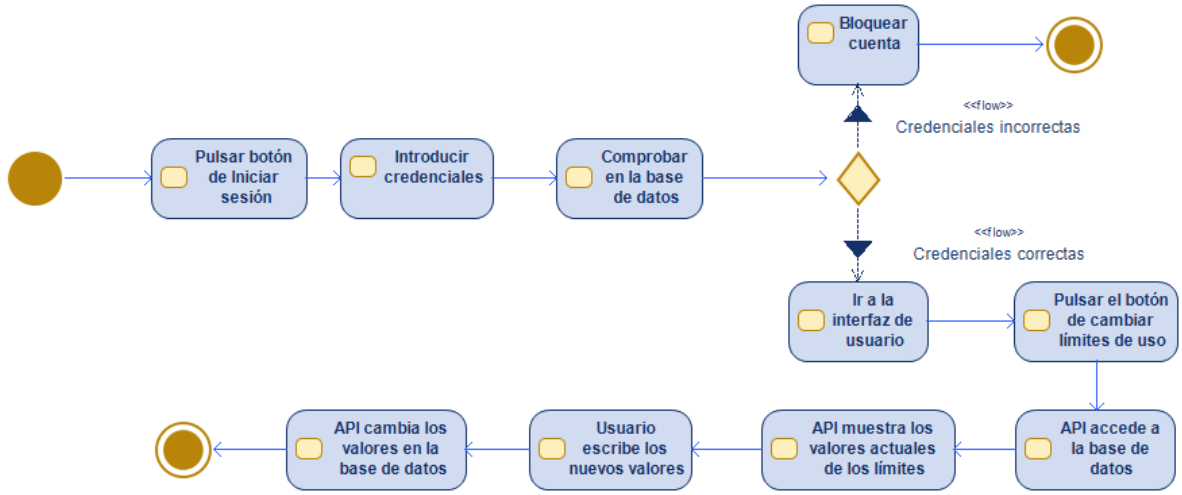
6.2 Diagrama de casos de uso



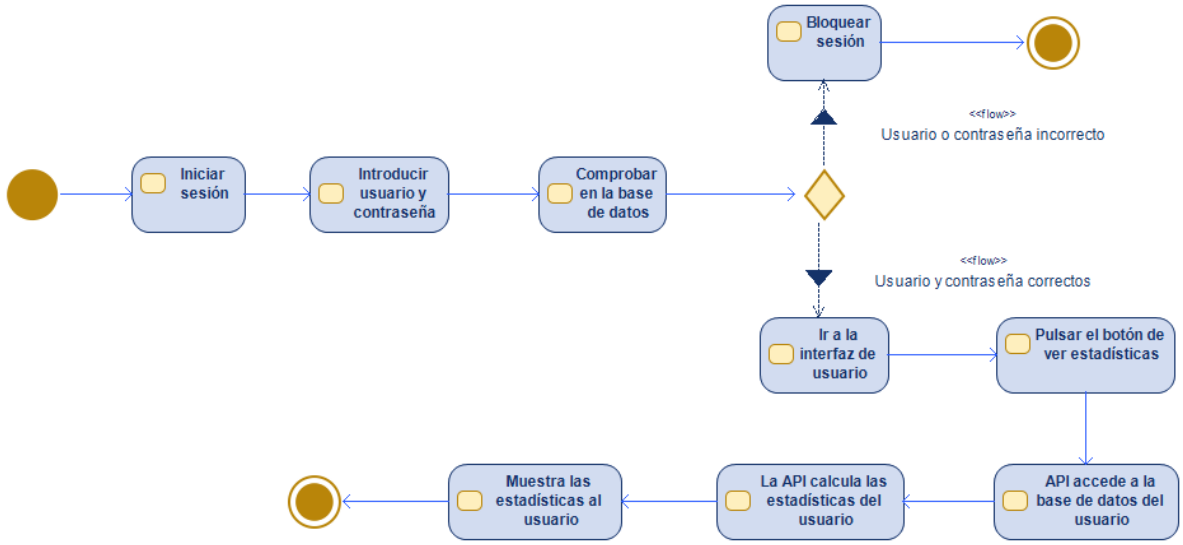
6.3 Diagramas de actividad

6.3.1 Educación y prevención

-Establecer límites de uso

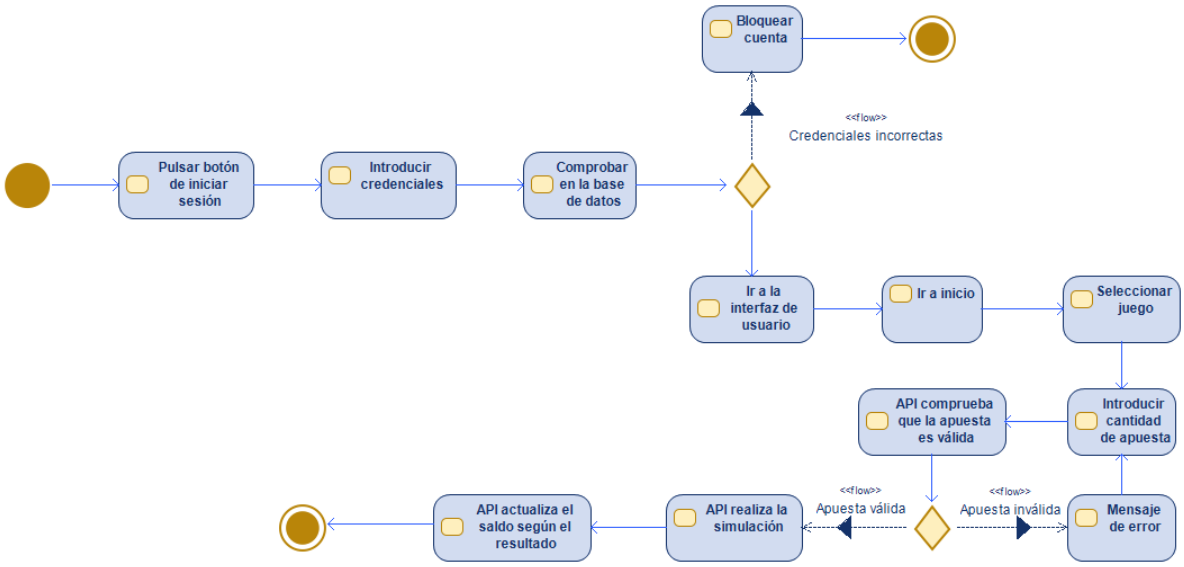


-Analizar comportamiento del usuario

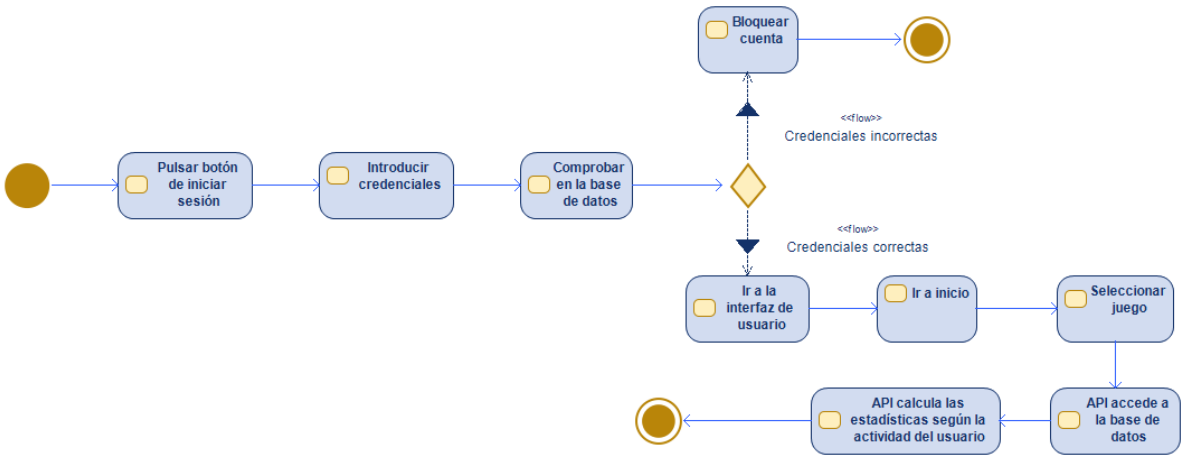


6.3.2 Simulaciones y juego

-Actualizar saldo

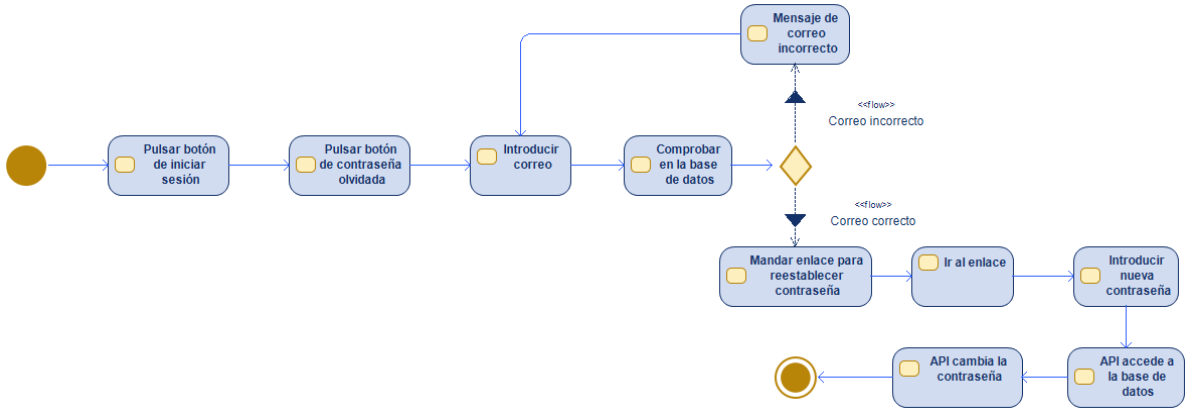


-Ver estadísticas en tiempo real

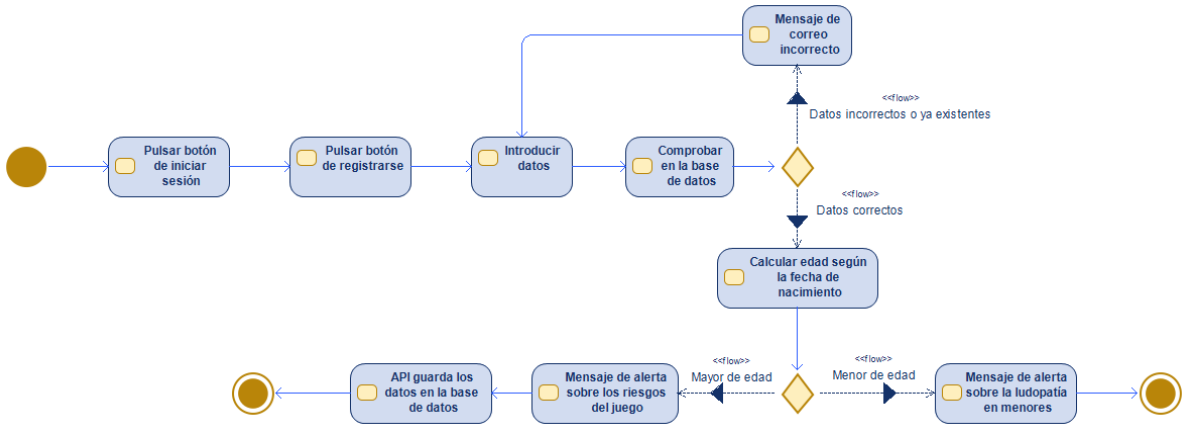


6.3.3 Autenticación

-Recuperar contraseña

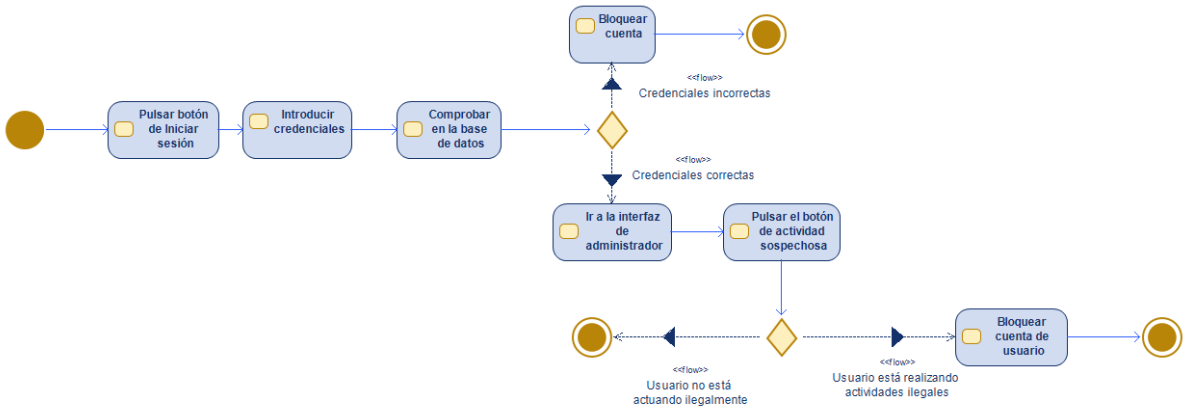


-Verificar edad

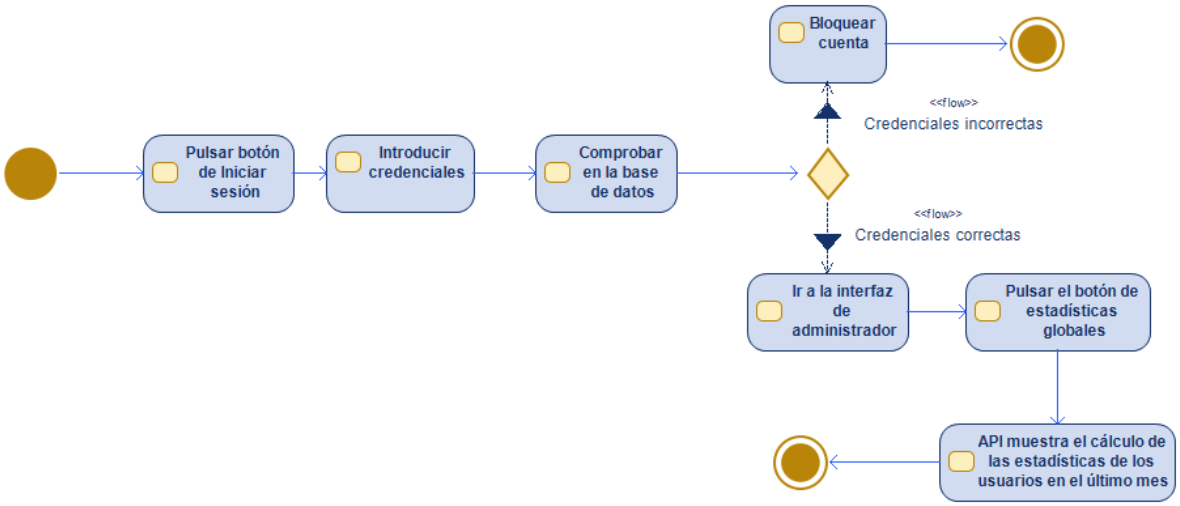


6.3.4 Gestión del sistema

-Bloquear cuenta



-Ver estadísticas globales



7. Plan de Proyecto

Hay que diseñar un plan de trabajo completo que cubra todas las fases del proyecto, desde que definimos los requisitos iniciales hasta el mantenimiento posterior al lanzamiento.

7.1. Fases del Proyecto

Fase I: Análisis Esta es la etapa de preparación. Antes de empezar a construir, vamos a dejar cerrado qué funcionalidades necesitamos y cuáles serán las fórmulas matemáticas y estadísticas que estarán en nuestro servicio externo (API). El objetivo es tener una base establecida y saber cómo se comportarán los cálculos de la API por dentro para no tener problemas más adelante.

Fase II: Diseño En esta etapa diseñamos la arquitectura BBDD y la interfaz visual (UX/UI). La idea es copiar la estética de los juegos de azar para que parezcan reales, pero añadiendo avisos e información por encima. Así, el usuario se encuentra un entorno familiar de casino, pero rodeado de gráficas y alertas que le avisan en tiempo real del riesgo de sus apuestas.

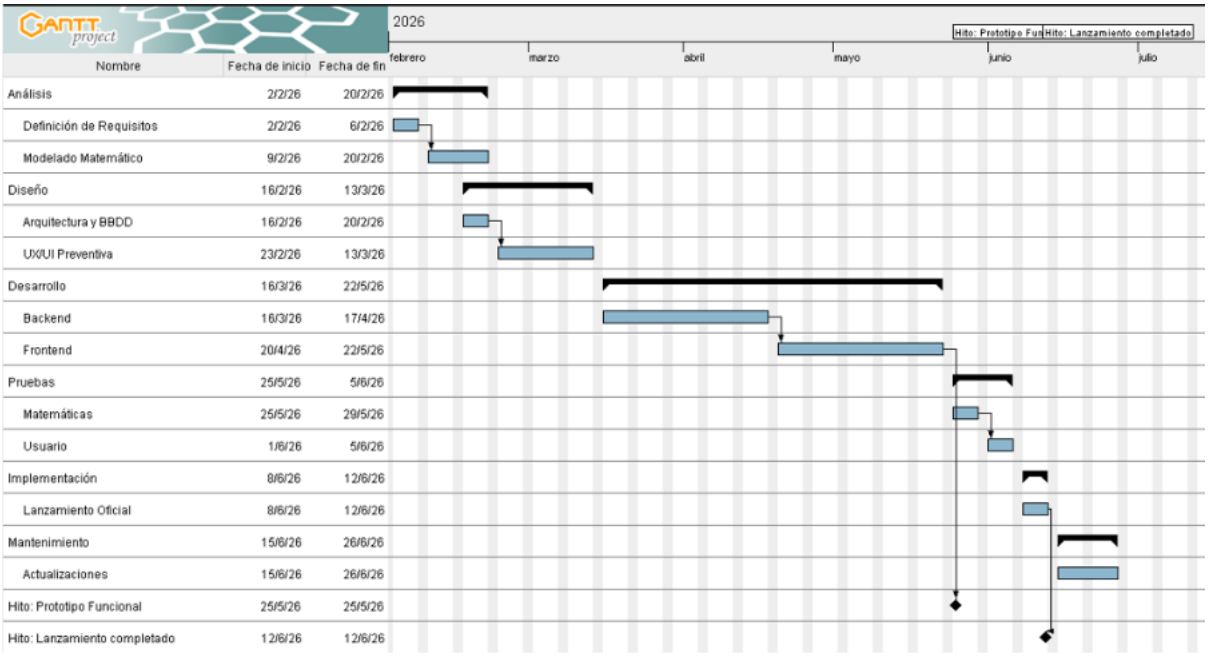
Fase III: Desarrollo Implementaremos el servicio externo (API) encargado de procesar los algoritmos matemáticos y estadísticos, y desarrollaremos el frontend (parte visual). El objetivo principal es conectar el juego con este servicio externo para que las gráficas reaccionen al instante; así el usuario verá cómo baja su saldo y entenderá visualmente que a la larga las probabilidades están en su contra.

Fase IV: Pruebas Antes de lanzar la aplicación, hay que probar que todo funcione bien. Haremos test técnicos para corregir errores, pero lo más importante es verificar que el servicio externo devuelve los cálculos correctos. Además, haremos sesiones con usuarios reales para comprobar no solo que el juego funciona, sino que la gente comprende las alertas y los datos educativos que les mostramos.

Fase V: Implementación Preparamos el lanzamiento de la aplicación. Esto implica configurar el servidor, activar los certificados de seguridad y subir la aplicación. Es el momento en que la versión 1.0 pasa de ser un desarrollo interno a estar accesible públicamente para los usuarios.

Fase VI: Mantenimiento El proyecto no termina con el lanzamiento. Nos basaremos en los reportes de los usuarios para lanzar actualizaciones periódicas. El objetivo no es solo arreglar posibles fallos técnicos, sino utilizar ese feedback real para mejorar la eficacia de la aplicación y asegurar que la herramienta evoluciona y sigue siendo útil.

7.2. Cronograma



Nombre	Fecha de inicio	Fecha de fin
Análisis	2/2/26	20/2/26
Definición de Requisitos	2/2/26	6/2/26
Modelado Matemático	9/2/26	20/2/26
Diseño	16/2/26	13/3/26
Arquitectura y BBDD	16/2/26	20/2/26
UX/UI Preventiva	23/2/26	13/3/26
Desarrollo	16/3/26	22/5/26
Backend	16/3/26	17/4/26
Frontend	20/4/26	22/5/26
Pruebas	25/5/26	5/6/26
Matemáticas	25/5/26	29/5/26
Usuario	1/6/26	5/6/26
Implementación	8/6/26	12/6/26
Lanzamiento Oficial	8/6/26	12/6/26
Mantenimiento	15/6/26	26/6/26
Actualizaciones	15/6/26	26/6/26
Hito: Prototipo Funcional	25/5/26	25/5/26
Hito: Lanzamiento completado	12/6/26	12/6/26

7.3. Dependencias e Hitos Clave

Para que el proyecto avance sin problemas, hay que establecer unas dependencias para no corromper el proceso, ya que algunas tareas dependen de otras, así como los hitos del proyecto.

Dependencias El trabajo sigue un orden lógico donde unas tareas desbloquean a otras:

- **Matemáticas y Servicio Externo:** Existe una dependencia total entre el análisis y la programación. No podemos empezar a programar la API del servicio externo si antes no hemos dejado cerradas las fórmulas matemáticas y estadísticas en la fase inicial.
- **API y Parte Visual:** Para que los simuladores funcionen, la parte visual (Frontend) necesita sacar los datos del servicio externo a través de la API. Por eso, aunque avancemos en el diseño, la parte visual no funcionará realmente hasta que este servicio esté operativo.

- **Pruebas y Lanzamiento:** La publicación de la app depende del éxito de las pruebas. No lanzaremos la plataforma al público hasta que confirmemos que

las matemáticas son exactas y que los usuarios entienden bien el mensaje preventivo.

Hitos Clave Estos son los dos momentos decisivos que nos servirán para saber si estamos cumpliendo los objetivos:

- **Prototipo Funcional:** Marca el final del desarrollo. En este momento ya tendremos una versión completa de la aplicación funcionando internamente. Los juegos y las gráficas ya estarán integrados con el servicio externo (API) y listos para empezar la fase de pruebas.
- **Lanzamiento Oficial v1.0:** Consiste en la publicación definitiva de la plataforma. Una vez confirmado que el sistema es seguro y el mensaje educativo eficaz, publicamos la herramienta en internet para que deje de ser un entorno de pruebas y empiece a funcionar como una plataforma de prevención real y accesible.

8. Recursos del Proyecto

En este apartado se detallan los recursos necesarios para el correcto desarrollo de la aplicación de prevención de la ludopatía. Se especifican los recursos humanos implicados, los recursos materiales y tecnológicos necesarios, así como una estimación aproximada del presupuesto requerido.

8.1 Equipo de trabajo: roles y responsabilidades

El proyecto será desarrollado por un grupo de estudiantes que se organizarán siguiendo una estructura descentralizada controlada con un jefe de proyecto y varios responsables de área para repartir mejor las tareas. De esta forma, se facilita la coordinación del trabajo y la colaboración entre todos los miembros del equipo.

A continuación, se describen los principales roles del equipo y las tareas que realiza cada uno:

Jefe de Proyecto

Es la persona encargada de organizar el trabajo general del grupo y asegurarse de que el proyecto avance correctamente. Sus principales funciones son:

- Organizar el desarrollo del proyecto y establecer un calendario de trabajo.
- Repartir las tareas entre los miembros del equipo.
- Comprobar que se cumplen los plazos establecidos.
- Resolver los problemas que puedan surgir durante el desarrollo.

- Mantener el contacto con el profesor o tutor del proyecto.
- Tomar las decisiones más importantes cuando sea necesario.

Este rol es importante para que todos los miembros del equipo sepan qué tienen que hacer y cuándo.

Analista de Requisitos

Se encarga de definir con claridad qué debe hacer la aplicación. Su trabajo consiste principalmente en:

- Recoger las necesidades del usuario.
- Analizar los requisitos funcionales y no funcionales.
- Elaborar los casos de uso del sistema.
- Definir las limitaciones y necesidades del usuario final.
- Dejar bien documentados los requisitos para que el equipo pueda trabajar con ellos.

Si los requisitos no están bien definidos desde el principio, el proyecto puede tener muchos problemas más adelante.

Diseñador UX/UI

El diseñador UX/UI se ocupa del diseño visual de la aplicación y de que sea fácil de usar. Entre sus tareas están:

- Diseñar las pantallas de la aplicación.
- Definir cómo se navega entre las distintas secciones.
- Adaptar el diseño a un público joven.
- Mejorar la experiencia del usuario para que la aplicación sea cómoda e intuitiva.
- Crear prototipos visuales antes de la implementación final.

Este rol es especialmente importante, ya que la aplicación debe ser atractiva y sencilla para los usuarios.

Desarrolladores

Son los encargados de programar la aplicación y hacer que todo funcione correctamente. Sus responsabilidades incluyen:

- Programar el frontend de la aplicación.
- Programar el backend y la lógica del sistema.
- Implementar el sistema de simulaciones de apuestas con moneda virtual.
- Gestionar el sistema de usuarios y la autenticación.
- Controlar el balance del usuario.

- Diseñar y gestionar la base de datos.
- Integrar todos los módulos del sistema.

Los desarrolladores trabajan de forma coordinada para que todas las partes de la aplicación funcionen correctamente.

Tester o Responsable de Calidad

Es la persona encargada de comprobar que la aplicación funcione bien antes de su entrega final. Sus principales tareas son:

- Preparar y realizar pruebas funcionales.
- Detectar errores en el sistema.
- Comprobar que se cumplen todos los requisitos.
- Verificar que las simulaciones funcionan correctamente.
- Revisar la estabilidad de la aplicación.

Gracias a este rol se reducen los fallos y se mejora la calidad del producto final.

8.2 Recursos materiales y tecnológicos

Para el desarrollo del proyecto es necesario contar con una serie de recursos materiales y tecnológicos que permitan diseñar, programar, probar y poner en marcha la aplicación.

Hardware

- Ordenadores personales para cada miembro del equipo.
- Dispositivos móviles (Android o iOS) para realizar pruebas.
- Un servidor de pruebas para alojar la aplicación durante el desarrollo.

Con estos recursos es suficiente para llevar a cabo el proyecto sin necesidad de una gran infraestructura.

Software

Se utilizarán herramientas habituales en proyectos de desarrollo de aplicaciones:

- Sistema operativo: Windows y Linux.
- Entornos de desarrollo: Visual Studio Code y Android Studio.
- Lenguajes de programación: Java, Kotlin, JavaScript o similares.
- Control de versiones: Git y GitHub para el trabajo en equipo.
- Base de datos: MySQL, Firebase o similar.
- Herramientas de diseño: Figma y Canva.

- Herramientas de gestión del proyecto: Trello, Jira o GitHub Projects.
- Herramientas de comunicación: Discord y Google Meet.

Estas herramientas facilitan la organización del trabajo y permiten que el equipo esté en contacto de forma rápida y continua.

Licencias

La mayor parte del software utilizado es gratuito o de código abierto. En los casos en los que sea necesario utilizar herramientas de pago, se recurrirá a versiones educativas, por lo que el coste será bajo.

8.3 Presupuesto estimado

Al tratarse de un proyecto académico, la mayoría de los recursos ya están disponibles para los estudiantes, por lo que el coste real del proyecto es bastante reducido. Aun así, se presenta una estimación aproximada:

Concepto	**Coste aproximado (€)**
Ordenadores personales	0 €
Dispositivos móviles	0 €
Servidor de pruebas	50 €
Licencias de software	0 – 100 €
Dominio web (opcional)	15 €
Total estimado	**65 – 165 €**

[Salto de ajuste de texto]Aunque no se tenga en cuenta económicamente, el mayor esfuerzo del proyecto es el tiempo que dedica cada integrante del equipo, ya que se deben realizar tareas de análisis, diseño, desarrollo, pruebas y documentación.

9. Gestión del Proyecto

9.1 Metodología

- Enfoque: Ágil — híbrido Scrum/Kanban.

- Cadencia:

- Sprints de 2 semanas con planificación, revisión y retrospectiva.

- Daily stand-up de 15 minutos.

- Artefactos:

- Product Backlog y Sprint Backlog derivados de RU/RF/RNF.

- Definition of Ready (criterios claros, estimación, dependencias).

- Definition of Done (pruebas, documentación mínima, verificación seguridad/privacidad, despliegue a entorno de pruebas).

- Flujo Kanban:

- Backlog → Ready → In Progress → Code Review → QA → Done.

- WIP limitado por columna para evitar sobrecarga.

9.2 Gestión de riesgos

- Proceso: identificación continua, matriz probabilidad/impacto, plan de mitigación, revisión semanal del registro.

- Riesgos iniciales y mitigaciones:

- GDPR (RNF15): minimización de datos, cifrado en tránsito y reposo, revisión legal.

- Contraseñas/seguridad (RNF01, RNF03): bcrypt/Argon2, bloqueo por intentos, auditoría de sesiones.

- Integridad de saldo (RNF12, RNF16): transacciones ACID, idempotencia, pruebas de resiliencia.

- Rendimiento simulaciones (RNF05): profiling, caching no sensible, presupuestos de rendimiento.

- Usabilidad/accesibilidad (RNF06, RNF08): tests con usuarios, auditoría WCAG 2.1 AA.

- Ética y prevención (RNF16, RNF17): límites (RF09), alertas educativas (RF10), revisión ética de contenidos.

- Compatibilidad (RNF09, RNF10): **pruebas cross-browser, responsive, matrices** en CI.

- Deriva de alcance: gobernanza de cambios y MoSCoW.

9.3 Control de calidad

- Revisión de código con checklist de seguridad y privacidad.

- Pruebas:

- Unitarias: cálculos de probabilidad/EV y lógica de límites.

- Integración: saldo virtual, historial, estadísticas en tiempo real.

- End-to-end: registro, simulación, pop-ups de 5 segundos.

- Accesibilidad: validación WCAG 2.1 AA.

- Auditorías: OWASP ASVS básico y Dependabot/SAST.

- Presupuestos de rendimiento: carga <3s, acciones <1s; monitorización en CI.

- Trazabilidad: RU/RF/RNF ↔ historias ↔ casos de prueba ↔ resultados; criterios de aceptación del punto 4.4 integrados en DoD.

9.4 Gestión de cambios

- Procedimiento (RFC):

1) Propuesta documentada (alcance, motivación, riesgos, criterios de éxito).

2) Revisión en backlog refinement.

3) Aprobación/Rechazo por Product Owner con priorización MoSCoW.

4) Implementación con actualización de documentación (SRS/UML).

5) Validación por QA y demostración.

- Criterios: evaluar impacto en seguridad, privacidad, rendimiento, alcance y cronograma.

9.5 Comunicación

- Reuniones:

- Daily, planificación, revisión y retrospectiva por sprint.
- Comité ético/seguridad mensual para RNF15–RNF17.

- Herramientas:

- Chat: Slack/Discord.
- Gestión de tareas: Jira/Trello con etiquetas RU/RF/RNF.
- Documentación: Confluence/Notion y README/SRS en repositorio.
- CI/CD: GitHub Actions con checks obligatorios.

- Reportes:

- Informe semanal de progreso y riesgos.
- Actas de decisiones y cambios aprobados.

10. Plan de Pruebas

El plan de pruebas garantizará que la plataforma de prevención y seguimiento de la ludopatía cumple los requerimientos funcionales y no funcionales definidos, asegurando especial cuidado en la fiabilidad de los cuestionarios de riesgo, la confidencialidad de los datos y la correcta gestión de alertas e intervenciones.

10.1 Tipos de pruebas

10.1.1 Pruebas unitarias

Se realizarán sobre los componentes individuales del sistema para detectar errores en etapas tempranas. Algunos ejemplos serían:

- Cálculo de puntuaciones en los cuestionarios de riesgo y su clasificación por niveles (bajo, medio, alto).
- Lógica de generación de alertas cuando se superan umbrales de riesgo.

- Validación de reglas de negocio (frecuencia de uso, patrones de juego, bloqueos temporales, etc.).

- Envío de notificaciones (a usuario, familiares o profesionales de apoyo) en distintos escenarios.

10.1.2 Pruebas de integración

Verificarán la correcta interacción entre módulos:

- Flujo completo: registro → cumplimentación de cuestionarios → cálculo de riesgo → generación de recomendaciones/alertas.

- Integración entre front-end, API y base de datos para el almacenamiento seguro de historiales de riesgo.

- Integración de módulos de estadísticas para profesionales (psicólogos, terapeutas) con los datos anonimizados de los usuarios.

- Integración con servicios externos (por ejemplo, proveedor de correo o SMS para avisos críticos).

10.1.3 Pruebas de sistema

Validarán el comportamiento global de la solución en un entorno lo más cercano posible al de producción:

- Escenarios típicos de usuario con riesgo emergente de ludopatía (uso intensivo en poco tiempo, respuestas de alto riesgo en cuestionarios).

- Gestión de distintos perfiles (usuario afectado, familiar/tutor, profesional sanitario).

- Comprobación de restricciones de acceso, confidencialidad de datos y trazabilidad de acciones.

10.1.4 Pruebas de aceptación (UAT)

Se llevarán a cabo con un grupo reducido de usuarios finales (personas en riesgo, familiares y profesionales) para validar que el sistema satisface sus necesidades:

- Validación de que el lenguaje de la interfaz y los mensajes es comprensible y no estigmatizante.
- Verificación de que el flujo de uso es sencillo incluso para usuarios con bajo nivel tecnológico.
- Confirmación de que las alertas y recomendaciones son útiles y accionables para la prevención de conductas de juego problemáticas.

10.1.5 Pruebas adicionales específicas

- **Pruebas de usabilidad:** sesiones con usuarios para medir facilidad de uso, tiempo para completar cuestionarios y comprensión de resultados.
- **Pruebas de seguridad:** verificación de cifrado de datos sensibles, gestión de sesiones, control de accesos y cumplimiento de RGPD.
- **Pruebas de rendimiento:** tiempo de respuesta en operaciones críticas (cálculo de riesgo, carga de historial, panel del profesional).

10.2 Estrategia y criterios de validación

Estrategia general

- Enfoque incremental e iterativo, alineado con la metodología de desarrollo definida en la Gestión del Proyecto.
- Definición de casos de prueba trazables a los requerimientos funcionales y no funcionales del SRS.
- Uso de entornos diferenciados: desarrollo, preproducción (para pruebas de integración/sistema) y producción.
- Uso de datos de prueba anonimizados que simulen distintos patrones de comportamiento de juego y niveles de riesgo.

Criterios de validación

Un requisito se considerará validado cuando se cumpla:

- Requerimientos funcionales:

  - Al menos un caso de prueba asociado por requisito.

  - 0 fallos abiertos de severidad crítica o alta.

  - Comportamiento observado coincide con la especificación (por ejemplo, un usuario de “alto riesgo” recibe siempre las alertas y recomendaciones correspondientes).

- Requerimientos no funcionales:

  -Tiempo de respuesta medio inferior al umbral definido (p.ej. < 2 segundos en cálculo de riesgo y carga de panel).

  - No se detectan vulnerabilidades críticas en pruebas de seguridad.

  - Niveles mínimos de satisfacción de usabilidad (p.ej. ≥ 80 % de usuarios piloto encuentra fácil entender su nivel de riesgo).

- Pruebas de aceptación:

  - Validación formal por parte de los representantes de usuario (por ejemplo, un psicólogo responsable y un grupo de usuarios/familiares).

  - Porcentaje mínimo de casos de uso clave superados (p.ej. ≥ 95 % de casos de uso prioritarios sin incidencias graves).

Los resultados de las pruebas se documentarán en informes de test, registrando incidencias, prioridad, estado y decisiones de corrección.

10.3 Herramientas de testing

Se utilizarán herramientas estándar de la industria, adaptadas a las tecnologías concretas del proyecto:

Pruebas unitarias e integración:

- Framework de pruebas del lenguaje/plataforma seleccionados (por ejemplo, JUnit / NUnit / Jest / PyTest).
- Mocking y stubs para simular servicios externos (envío de correo/SMS, pasarelas, etc.).

Pruebas de API:

- Postman o herramienta equivalente para validar endpoints relacionados con cuestionarios, perfiles de riesgo, alertas y paneles de profesionales.

- Ejecución automática de colecciones de pruebas (Newman o integración similar en CI).

Pruebas end-to-end (E2E) y de interfaz:

- Herramientas como Cypress / Selenium WebDriver para automatizar escenarios críticos de usuario (registro, cumplimentación de cuestionarios, consulta de recomendaciones).

Integración continua y calidad:

- Pipeline de Integración Continua (por ejemplo, GitHub Actions, GitLab CI o similar) para ejecutar automáticamente suites de pruebas en cada commit.
- Herramientas de análisis estático de código y cobertura de pruebas (por ejemplo, SonarQube o equivalentes).

Estas herramientas permitirán garantizar que la solución de apoyo a la prevención y tratamiento de la ludopatía sea robusta, segura y fiable antes de su despliegue a producción.

Casos de uso:Casos de uso (1).docx

Prototipo:prototipo.html