Compositor de Software: Juego de detectives

Founderz | Microsoft

01 Juego de detectives impulsado por IA

Concepto general del juego de detectives

El proyecto consiste en utilizar inteligencia artificial generativa para componer el código de una aplicación web que recrea un juego de detectives al estilo de Carmen Sandiego.

Inspiración y objetivo del juego

El punto de partida es el clásico juego de investigación **"¿Dónde está Carmen Sandiego?"**, en el que una persona jugadora persigue a un ladrón viajando por distintas ciudades del mundo. Para avanzar es necesario interpretar **pistas sobre geografía, cultura e historia** hasta identificar el siguiente destino.

El objetivo del nuevo juego es mantener ese espíritu detectivesco, pero con una implementación web sencilla, atractiva y adecuada incluso para un **público infantil**, de forma que el reto se base en escoger entre varias opciones y no en escribir la respuesta exacta.

Requisitos funcionales de la aplicación web

La especificación del proyecto se traslada a la IA mediante un prompt detallado que define los **requisitos funcionales** del juego. Entre ellos destacan:

- Presentar un **escenario inicial de robo** en una ciudad famosa, que sirva como introducción narrativa.
- Ofrecer **tres pistas** sobre el siguiente destino, relacionadas con geografía, cultura e historia.

- Permitir elegir entre **tres posibles ciudades** basadas en esas pistas, para facilitar el juego a personas que aún no dominan la respuesta abierta.
- Incluir al menos **cinco ciudades diferentes** durante la persecución, de forma secuencial.
- Implementar un **sistema de puntos** que recompense las respuestas correctas.
- Incorporar un **final con varios sospechosos**, donde haya que identificar al ladrón.
- Ofrecer una **interfaz atractiva y responsiva**, que se adapte a distintos tamaños de pantalla.
- Concentrar todo en **un único archivo HTML**, utilizando Tailwind desde un **CDN** para los estilos.

Papel de la inteligencia artificial generativa

Modelos como **ChatGPT** o **Copilot**, incluso en sus versiones gratuitas, permiten generar de una sola vez el esqueleto completo de la aplicación: estructura HTML, estilos con Tailwind y lógica JavaScript integrados en un mismo archivo.

La persona que diseña el juego actúa como **"compositora de software"**: no escribe todo el código desde cero, sino que especifica qué debe hacer el programa, revisa el resultado y guía a la IA con correcciones sucesivas hasta lograr una experiencia jugable coherente y atractiva.

Flujo básico de una partida

El flujo del juego se estructura en varias etapas consecutivas que la IA debe reflejar en el código generado.

1. **Presentación del caso**: La aplicación muestra un mensaje de alerta indicando que se ha producido un robo en una ciudad famosa y que el objetivo es atrapar al ladrón siguiendo las pistas.

2. **Lectura de las pistas**: En cada ronda se ofrecen tres pistas relacionadas con geografía, cultura e historia. Ejemplos: existencia de un gran parque urbano, playas conocidas o un monumento emblemático.

3. **Selección de la ciudad**: Se presentan tres posibles destinos. Quien juega selecciona una de las opciones en función de la interpretación de las pistas, sin conocer aún si es correcta.

4 **Evaluación y puntuación**: El juego comprueba la respuesta y muestra un mensaje de acierto o error, actualizando la **puntuación acumulada**. La lógica de puntos se puede simplificar o hacer más exigente según el diseño.

5 **Progresión por distintas ciudades**: Tras cada acierto, el ladrón huye a una nueva ciudad. El juego repite el ciclo de pistas, opciones y verificación durante al menos cinco localizaciones diferentes.

6 **Fase final: elección del ladrón**: Una vez completado el recorrido por las ciudades, la aplicación muestra varios sospechosos. A partir de la información vista durante la partida, se debe seleccionar quién es el ladrón y se revela la **puntuación final**.

Componentes de aprendizaje integrados en el juego

El diseño del juego no solo busca entretenimiento, sino también fomentar conocimientos y habilidades mediante su mecánica.

Geografía aplicada

Las pistas obligan a relacionar ciudades con elementos geográficos concretos, como grandes parques urbanos, ríos, playas o localización continental.

Cultura y estilo de vida

Referencias a gastronomía, festivales o formas de ocio (queso y vino, carnavales, playas famosas) permiten conectar cultura local y ciudad correspondiente.

Historia y acontecimientos clave

Hechos históricos y eventos internacionales, como unos Juegos Olímpicos recientes, sirven como pistas para deducir el destino correcto.

Pensamiento deductivo y motivación

El sistema de opciones múltiples y puntos favorece el razonamiento lógico, el ensayo y error controlado y la motivación a través del progreso.

Cuanto más claros y detallados sean los requisitos iniciales del juego (mecánicas, narrativa y estética), más alineado estará el código generado por la IA con el diseño deseado.

02 Refinar el código junto a la IA

Colaboración iterativa con la IA generativa

El valor de la IA en este tipo de proyectos se maximiza cuando se entra en un ciclo iterativo de prueba, detección de errores y correcciones bien formuladas.

Primer resultado: cuando el juego "funciona a medias"

En la primera versión generada, el enunciado del juego indicaba directamente el nombre de la ciudad: por ejemplo, "Se ha cometido un robo en París". De esta manera, el sistema estaba **revelando la respuesta** antes incluso de mostrar las pistas, lo que eliminaba por completo el misterio.

Este tipo de situaciones ilustra la importancia de **probar el juego generado** y evaluar si las decisiones narrativas de la IA tienen sentido desde el punto de vista de la experiencia de juego.

Solicitar correcciones específicas y detalladas

Para corregir el problema se envió un mensaje claro a la IA, destacando que resultaba absurdo dar la solución en el propio enunciado y pidiendo que se **reformulase la introducción.** Además, se añadieron peticiones de mejora estética: hacer el juego más visual, añadir **emojis en las opciones**, incluir emojis grandes de ladrón o detective y utilizar **degradados y fuentes más llamativas.**

Cuanto más concretas son las indicaciones (qué cambiar, por qué y cómo debería verse el resultado), más fácil resulta que la IA produzca una nueva versión que encaje con el objetivo del diseño.

Gestión de generaciones largas de código

Cuando el archivo HTML se hace más extenso, el modelo puede quedarse a medias y no generar todo el código en una sola respuesta. Es fundamental comprobar que el resultado incluye el cierre de las etiquetas **</body>** y **</html>**.

En caso de corte, se puede utilizar el botón de **"Continuar generando"**. De este modo, el resto del código se completa en la misma ventana, lo que permite copiarlo en un solo bloque. Como alternativa, es posible escribir "Continúa", pero esto suele producir una segunda respuesta separada que obliga a combinar manualmente ambas partes.

Control sencillo de versiones del archivo

Para no perder el trabajo previo, resulta práctico guardar el código en distintos archivos, por ejemplo "Juego de Detectives v1" y "Juego de Detectives v2". Así se conserva un **historial de versiones** que permite comparar tanto el aspecto visual como la estructura del código.

En el caso descrito, la longitud del archivo pasó de unas 170 líneas a aproximadamente 184, reflejando la introducción de mejoras visuales y de usabilidad sin necesidad de rehacer el proyecto desde cero.

Versión inicial frente a versión mejorada

La comparación entre versiones ayuda a entender cómo influyen las indicaciones dadas a la IA en el resultado final.

Aspecto	Versión inicial	Versión mejorada
Título y narrativa	Título simple y texto introductorio que llegaba a revelar la ciudad de forma directa.	Título más atractivo ("Juego de Detectives: ¡Encuentra al ladrón!") y enunciado que mantiene el misterio.
Gestión del misterio	El enunciado mencionaba explícitamente ciudades como París, anulando el reto de deducción.	Las ciudades se deducen solo a partir de las pistas de geografía, cultura e historia.
Estética visual	Diseño web muy básico, con pocos elementos visuales llamativos.	Uso de colores degradados, fuentes más modernas y **emojis** que refuerzan la temática detectivesca.
Interacción y feedback	Mensajes funcionales pero poco expresivos al acertar o fallar.	Mensajes más cuidados, con feedback claro sobre aciertos, progresión y resultado final.
Estructura del código	Archivo algo más corto, con la funcionalidad básica implementada.	Archivo ligeramente más largo, incorporando mejoras de interfaz sin complicar en exceso la lógica.
Claridad de la experiencia	La intención del juego se entiende, pero la ejecución narrativa genera confusión.	La experiencia resulta más coherente: presentación, pistas, opciones, puntos y fase final de sospechosos.

Proceso de refinamiento del juego con IA

El desarrollo del juego sigue un ciclo de mejora continua guiado por la interacción entre la persona diseñadora y la IA.

1. **Definir el prompt inicial**: Se redacta una instrucción detallada que incluya el tipo de juego, las mecánicas principales, el número de ciudades, el sistema de puntos y los requisitos técnicos (archivo único, Tailwind desde CDN, diseño responsivo).

2. **Generar y revisar la primera versión**: La IA produce el archivo HTML completo. A continuación se revisan la narrativa, la jugabilidad y la interfaz para detectar incoherencias o aspectos mejorables.

3. **Identificar problemas clave**: Se señalan errores concretos, como revelar la ciudad en el enunciado, o carencias, como un diseño poco visual o mensajes poco claros.

4 **Redactar peticiones de cambio precisas**: Se explica qué debe cambiarse y cómo. Ejemplos: ocultar la ciudad en la introducción, añadir emojis grandes de ladrón y detective, o aplicar fondos degradados.

5 **Obtener una nueva versión y compararla**: La IA genera una versión corregida que se guarda con otro nombre. Se comparan apariencia, claridad del juego y longitud del código respecto a la anterior.

6 **Iterar según sea necesario**: El proceso puede repetirse tantas veces como haga falta para ajustar detalles de jugabilidad, estética o contenido sin perder el control sobre el propósito del juego.

i La IA no tiene por qué acertar a la primera; su fortaleza está en acelerar ciclos de prueba y mejora cuando recibe comentarios concretos y estructurados.

03 Personalizar y ampliar el juego

Calidad de las pistas y evolución del diseño

Una vez que la estructura del juego funciona, resulta clave cuidar la precisión de las pistas, la coherencia del contenido y las posibles extensiones.

Verificación y mejora de las pistas

Las pistas generadas por la IA pueden contener **inexactitudes**. Por ejemplo, describir Tokio como "la ciudad más poblada del mundo" puede generar dudas, aunque sí tenga sentido mencionar que allí se celebraron unos Juegos Olímpicos recientes.

En estos casos se puede ajustar el texto de la pista para que sea más riguroso, por ejemplo cambiando a "es la capital del país donde nació el anime". La revisión humana garantiza que el contenido sea **fiable y pedagógicamente sólido**.

Edición manual del código y sincronización con la IA

El código generado sirve como base editable. Es posible modificar directamente el archivo HTML para retocar pistas, corregir datos o ajustar la lógica de puntuación. Sin embargo, la IA no conoce esos cambios a menos que se le proporcione la versión actualizada.

Para mantener la **sincronización**, resulta útil arrastrar y soltar el archivo HTML modificado en la conversación con la IA antes de pedir nuevos ajustes. Así, el modelo trabaja siempre sobre el estado más reciente del proyecto y evita reintroducir errores ya corregidos.

Ajuste de puntuación y mensajes finales

La mecánica de puntos del juego puede evolucionar fácilmente. A partir de la base ya generada se pueden introducir, por ejemplo, **penalizaciones por respuestas incorrectas** para incrementar el reto o matizar los mensajes mostrados.

También se pueden pulir detalles de lenguaje, como adaptar el texto a singular o plural ("has ganado un punto" frente a "has ganado dos puntos") o añadir más información narrativa sobre los sospechosos para que la elección final resulte más significativa.

Líneas de evolución posibles del juego

Partiendo del prototipo básico, el juego puede crecer en profundidad y riqueza sin perder su sencillez técnica.

Pistas más ricas y variadas

Añadir descripciones culturales, históricas y geográficas adicionales que mantengan el misterio sin revelar la ciudad de forma directa.

Reglas de puntuación más elaboradas

Introducir bonificaciones por rachas de aciertos y penalizaciones por errores para modular el nivel de dificultad.

Experiencia visual reforzada

Incorporar más emojis temáticos, cambios de color según acierto o fallo y pequeños efectos visuales que acompañen la historia.

Ampliación de contenido

Aumentar el número de ciudades, añadir nuevos sospechosos y ofrecer datos curiosos adicionales sobre cada localización visitada.

El archivo creado con ayuda de la IA actúa como un prototipo sobre el que practicar diseño de experiencias, programación básica y revisión crítica de contenidos.

Creado por Victoria, AI Founderz Fellow, y aprobado por el equipo de Founderz.

Última actualización 5 de diciembre de 2025

Este documento fue originalmente generado por la IA y revisado por nuestro equipo humano. En Founderz, utilizamos la IA de forma responsable y transparente.