# PROYECTO FINAL DE INGENIERÍA

# GAME PROTOGEN: INTELIGENCIA ARTIFICIAL GENERATIVA EN EL PROTOTIPADO DE VIDEOJUEGOS EN 2025

**Autor/es:**

Fain Fernández, Máximo Alfredo - LU: 1143649

**Carrera:**
Ingeniería en Informática

**Tutor/es:**
Prieto, Leonardo Javier

**Año:**
2025

Universidad Argentina de la Empresa
Facultad de Ingeniería y Ciencias Exactas

UADE

**PROYECTO FINAL DE INGENIERÍA**

# GAME PROTOGEN: INTELIGENCIA ARTIFICIAL GENERATIVA EN EL PROTOTIPADO DE VIDEOJUEGOS EN 2025

**Fain Fernández, Máximo Alfredo – LU114349**

Ingeniería en Informática

Tutor:
**Prieto, Leonardo Javier, UADE**

**2025**

**UNIVERSIDAD ARGENTINA DE LA EMPRESA**

FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS

## Agradecimientos

Quiero empezar este apartado agradeciendo a mi familia. Principalmente a mi mamá Miriam G. Fernández y a mi papá Claudio G. Fain. Ellos me contuvieron durante todo el año en el que se desarrolló este proyecto pese a días en los que estuve desganado. También a mi hermano Lautaro E. Fain Fernández, a quien aprecio mucho y me dió muchas veces la confianza que necesitaba para seguir adelante.

Por otro lado, también quiero darle las gracias a mis amigos Seba, Herni, Juan, Joa y Bauty, con los cuales pude descansar de tanto trabajo en hermosas y divertidas noches jugando videojuegos.

Además, agradecer a Nicolás Irigoyen y Timoteo Lombardo, dos grandes y maravillosas personas que conocí en mi transcurso por la universidad y puedo llamar amigos. Gracias a ellos, estos años de estudio fueron mucho más amenos.

En adición, darle las gracias a maravillosos profesores que me transmitieron su pasión durante mi paso por la universidad. Entre ellos, Ricardo A. Wehbe, Claudia G. Lopez, Anibal Freijo, Charles Maldonado.

Asimismo, me gustaría dar una mención importante a mi Tutor, Leonardo Prieto. Sin su ayuda no podría haber completado este trabajo.

Por último, pero no menos importante, quiero agradecerme a mi. Muchas veces durante el transcurso del año tuve todo en contra. Dolores de cabeza, semanas sin salir de mi casa, entre otras cosas. A pesar de todo eso, no me rendí y seguí adelante por más de que mi cabeza me pedía que pare.

Para finalizar este apartado, me gustaría citar una frase de una saga de videojuegos que me representa firmemente. Sin más que añadir, *SIC PARVIS MAGNA*.

## Resumen

En los últimos años, la etapa de prototipado en el desarrollo de videojuegos ha recibido una atención comparativamente menor que las fases de producción y pulido, pese a su impacto directo en tiempos, costos y calidad final. En línea con trabajos que priorizan acortar la iteración temprana, en el presente proyecto el objetivo fue diseñar y construir Game Protogen, un motor orientado al prototipado de videojuegos 2D de plataformas con IA generativa integrada. Para ello, se implementó un editor  y un orquestador multi-agente que transforma instrucciones en lenguaje natural en operaciones sobre la escena (generación de sprites, texturas y código), junto con exportación a ejecutable jugable. La investigación con usuarios evidencia la necesidad de herramientas específicas para acelerar la preproducción. Los resultados indican que focalizar la IA en el prototipado permite acortar ciclos, reducir costos y mejorar la toma de decisiones antes de la producción completa, sentando bases para su extensión a otros géneros y flujos de trabajo.

**Palabras clave:** *prototipado de videojuegos; IA generativa; videojuegos de plataformas 2D; orquestación multi-agente; motor de videojuegos*

## Abstract

In recent years, the prototyping stage in game development has received comparatively less systematic attention than production and polishing, despite its direct impact on timelines, costs, and final quality. Aligned with work that prioritizes shortening early iteration, this project set out to design and build Game Protogen, a 2D platformer–oriented game-prototyping engine with integrated generative AI. To this end, an editor and a multi-agent orchestrator were implemented to translate natural-language instructions into scene operations (generation of sprites, textures, and code), together with export to a playable executable. User research underscores the need for dedicated tools to accelerate pre-production. The results indicate that focusing AI on prototyping shortens cycles, reduces costs, and improves decision-making prior to full-scale production, laying the groundwork for extension to other genres and workflows.

**Keywords:** *game prototyping; generative AI; 2D platformer games; multi-agent orchestration; game engine.*

# Contenidos

# 1. Introducción

En los últimos años, la industria de los videojuegos ha crecido rápidamente. Tal es este crecimiento que según NewZoo, se estima que para 2029 la industria tendrá ingresos de 92.7 mil millones de dólares («The PC & Console Gaming Report 2025», 2025). Una cifra similar a una predicción consultada de Statista sobre la industria del cine, que proyecta un valor de mercado de 104,35 mil millones de dólares estadounidenses para 2029 («Cinema - Worldwide», 2025). Por lo tanto, no es difícil notar que los videojuegos son una gran oportunidad en la cual invertir.

Ahora bien, los videojuegos son un producto, y, como tal, tienen su propio proceso de desarrollo. Una de las cuestiones esenciales es el prototipado. Como definición, "un prototipo de juego es una versión temprana y simplificada de un juego que se utiliza para probar y experimentar. [...] La fidelidad y complejidad pueden variar drásticamente. La clave es probar e iterar rápidamente sobre los elementos más críticos del juego" (Parente, 2023). A su vez, "el prototipado sirve como la columna vertebral de cualquier proyecto exitoso de videojuegos. Permite a los desarrolladores crear una representación tangible de sus ideas y explorar diversas posibilidades de diseño antes de invertir recursos significativos en el desarrollo a gran escala" (Andersen, 2024). Sin embargo, como dice James Margaris, un desarrollador experimentado, el prototipado prolongado puede ser una pérdida de tiempo. James menciona que los prototipos deberían ser, en teoría, baratos y rápidos. No obstante, en la práctica, estos se vuelven una actividad común en la fase de desarrollo y requieren de la participación de personal senior, lo que resulta costoso y, muchas veces, prolongado (Margaris, 2025). De esta manera, es sensato pensar en resolver esta problemática utilizando herramientas actuales como la inteligencia artificial.

Por lo tanto, frente a esta necesidad de agilizar y abaratar la etapa de prototipado y viendo la oportunidad en el mercado, se plantea una solución mediante el uso de inteligencia artificial generativa integrada en un motor de videojuegos especializado en el prototipado del producto. El propósito es que desarrolladores de videojuegos puedan probar y validar mecánicas de videojuegos previo al embarque en el desarrollo completo del mismo de manera simple y rápida.

## 1.1. Objetivos

Desarrollar un Motor de Videojuegos integrado con IA Generativa para la creación de prototipos en Argentina, en el año 2025.

Para alcanzar este objetivo, se tienen como objetivos específicos:

- Entrevistar al menos dos personas dentro de la industria de los videojuegos y encuestar por lo menos a 50 desarrolladores de videojuegos para validar la solución.
- Desarrollar un motor de videojuegos especializado en prototipado.
- Utilizar una inteligencia artificial generativa que permita la creación de al menos tres tipos de activos para los prototipos de videojuegos: texturas, sprites y código.
- Desplegar una infraestructura en la nube para garantizar escalabilidad, alta disponibilidad y seguridad.

## 1.2. Alcance

El producto es un motor de videojuegos integrado con Inteligencia artificial generativa y especializado en el prototipado para escritorio en Windows 10.

El motor está enfocado en videojuegos 2D de plataformas. Al estar orientado en prototipos de videojuegos, es de esperar que algunas partes de estos prototipos no funcionen de manera correcta. Pero es mediante la iteración con la IA Generativa que se focaliza en mejorarlas.

Además, el sistema cuenta con:

- Herramientas básicas de manipulación: escalar, mover y rotar entidades
- Chat integrado con IA a través del cual generar contenido del prototipo.
- Exportar prototipo como ejecutable jugable.
- Editor visual, mediante el cual interactuar con las herramientas y el prototipo
- Cuentas de usuario que identifican a las personas que utilicen la herramienta.

No obstante, quedan fuera del alcance del prototipo funcional final:

- Enfoque del motor de videojuegos en otros géneros.
- Gráficos en 3D.

- Configuraciones avanzadas.
- Árbol de entidades.
- Sistema de físicas complejo.
- Sistema de audio.

# 2. Antecedentes

Con el auge de la Inteligencia Artificial, cada vez se ven más aplicaciones de esta tecnología a diversas industrias y rubros. En el presente documento se explora su aplicación al prototipado de videojuegos, aunque previamente se requiere una revisión de los conceptos relevantes y del estado actual del conocimiento en la materia.

## 2.1. Marco Teórico

En esta sección se procede a un repaso de algunos conceptos fundamentales involucrados a lo largo del trabajo. En primera instancia, se desarrollan conceptos de inteligencia artificial. En segunda instancia, se abordan nociones particulares del desarrollo y prototipado de videojuegos.

### 2.1.1. Inteligencia Artificial

La Inteligencia Artificial (IA) es un concepto que se ha definido de distintas formas a lo largo de la historia. Por ejemplo, Bellman decía que refiere a "*la automatización de actividades que vinculamos con procesos de pensamiento humano, actividades como la toma de decisiones, resolución de problemas, aprendizaje.*" (Bellman, 1978). Por otro lado, Kurzweil comenta que la inteligencia artificial es "*El arte de desarrollar máquinas con capacidad para realizar funciones que cuando son realizadas por personas requieren de inteligencia*" (Kurzweil, 1990). Optando por una definición más moderna, Google presenta que "*La inteligencia artificial (IA) es un conjunto de tecnologías que permiten que las computadoras realicen una variedad de funciones avanzadas, incluida la capacidad de ver, comprender y traducir lenguaje hablado y escrito, analizar datos, hacer recomendaciones y mucho más.*" («¿Qué es la inteligencia artificial o IA?», [sin fecha])

En definitiva, se puede decir que la inteligencia artificial busca dotar a las máquinas de inteligencia y racionalidad, características humanas.

#### 2.1.1.1. Machine Learning

Machine learning (ML) o aprendizaje automático es una de las ramas de la inteligencia artificial. Se trata del conjunto de métodos utilizados para extraer patrones de los datos y, luego, con esa información realizar predicciones o tomar decisiones (Murphy, 2012). Como su título lo indica, lo que se busca es conceder a las máquinas la habilidad de aprender sin haber sido explícitamente programadas para ello (Samuel, 1959).

En particular, entre varios tipos de aprendizaje se destacan:

- Aprendizaje Supervisado: consiste en aprender de las entradas y las salidas (Russell y Norvig, 2004). Se utiliza para problemas de clasificación y predicciones.
- Aprendizaje No Supervisado: se basa en "*aprender a partir de patrones de entradas para los que no se especifican los valores de sus salidas*" (Russell y Norvig, 2004). Aplicado a reconocimiento de tendencias y agrupamiento.
- Aprendizaje Por Refuerzo: el algoritmo de aprendizaje debe descubrir por sí mismo las entradas y salidas a través de un proceso de prueba y error (Bishop, 2006). Entonces, cuando la máquina (o agente) toma buenas decisiones, se le dan recompensas, y cuando toma malas decisiones, penalizaciones. Es por esto que este tipo de aprendizaje se usa en donde las acciones tienen consecuencias a largo plazo y el entorno presenta un grado de incertidumbre.
- Aprendizaje Autosupervisado: En inglés, Self-Supervised Learning (SSL), se centra en generar etiquetas de salida "intrínsecas" a las entradas, revelando posibles relaciones (Gui et al., 2024). Es decir, estas etiquetas de salida son derivadas directamente de las entradas. Su principal utilidad radica en la capacidad de extraer representaciones útiles y generalizables a partir de datos no etiquetados.

#### 2.1.1.2. Redes Neuronales Artificiales

Las redes neuronales son un modelo de ML que busca ser una representación artificial del funcionamiento del cerebro humano, ya que se piensa que la capacidad de procesamiento de información del cerebro radica en las neuronas (Russell y Norvig, 2004). Una red neuronal artificial está compuesta de unidades, que se llaman neuronas artificiales, las cuales están organizadas en capas. Su arquitectura define cómo estas unidades se conectan entre sí (Braga-Neto, 2020). Cada conexión en la red tiene un peso que determina la fuerza de la conexión (Russell y Norvig, 2004) de tal manera que a la hora del entrenamiento estos pesos se ajustan, lo que se traduce en una mayor capacidad de generalización. Se suelen estructurar en forma de capas en donde la salida de una es la entrada de la siguiente. La capa que recibe los datos se llama **Capa de Entrada**, esta se conecta con la **Capa Oculta**, en donde se procesan y transforman la información para finalmente enviarla a la **Capa de Salida** que genera el resultado final. Además, es usual agregar capas ocultas aumentando las neuronas artificiales, cuya ventaja es ampliar el espacio de hipótesis (Russell y Norvig, 2004) permitiéndole a la red aproximar funciones de entrada-salida más complejas. La utilidad de las redes neuronales radica en que son capaces de resolver problemas que los programadores no podrían debido a la cantidad de casos particulares que estos contienen (Berzal, 2018).

Existen dos tipos principales de arquitecturas en las redes neuronales:

- Redes con alimentación hacia delante: representa una función de sus entradas actuales, es decir, no mantiene estado interno (Russell y Norvig, 2004). Se puede decir que son acíclicas, la información fluye en una sola dirección, desde la capa de entrada hasta la capa de salida. Utilizada para problemas de clasificación y regresión, entre otros.
- Redes recurrentes: permite que sus salidas alimenten sus entradas, de manera tal que este tipo de redes, mantienen memoria a corto plazo (Russell y Norvig, 2004). En este caso, se habla de redes cíclicas. Comúnmente aplicada al procesamiento de lenguaje natural.

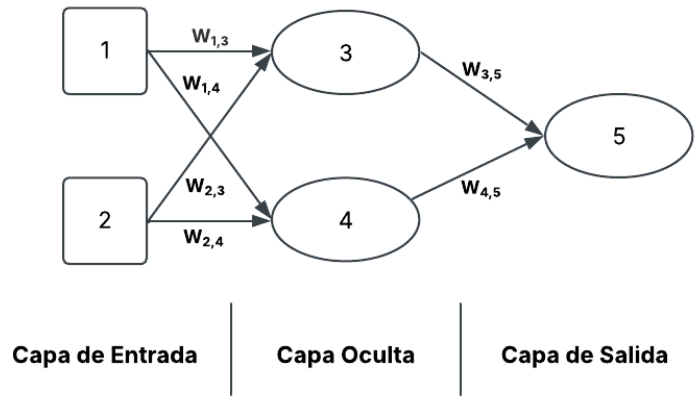


Figura 1: red neuronal sencilla, dos entradas, una capa oculta de dos unidades y una salida (Russell y Norvig, 2004).

### 2.1.1.3. Deep Learning

Aprendizaje profundo o Deep Learning (DL) refiere a una red neuronal en donde hay muchas capas en la capa oculta, y a su vez, existen muchas neuronas en ellas (Goodfellow et al., 2016). Los modelos de Deep Learning proporcionan herramientas que permiten construir características a partir de otras características, y así, construir modelos jerárquicos de estas y múltiples niveles de abstracción que dejan de lado detalles irrelevantes y se centran en factores clave (Berzal, 2018). Por ejemplo, en un modelo de Aprendizaje Profundo que trabaja con imágenes, la red, al comienzo, es capaz de identificar los bordes de la imagen, y en capas posteriores, la red busca combinar fronteras para identificar la presencia de objetos independientemente si este aparece menos iluminado en la imagen o sutilmente inclinado en alguna dirección.

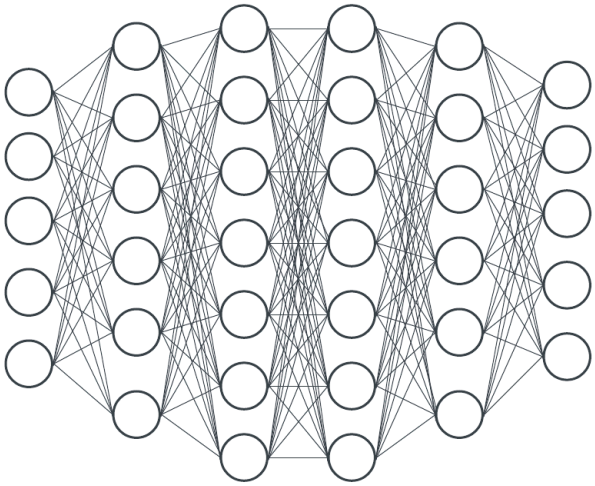

Figura 2: red neuronal profunda. (Gráfico de elaboración propia, 2025).

#### 2.1.1.4. LLMs

Los modelos de lenguaje de gran escala o Large Language Models (LLMs) son redes neuronales profundas basadas en la arquitectura Transformer (Vaswani et al., 2023). A diferencia de las arquitecturas recurrentes, esta arquitectura es más paralelizable, logrando una mayor eficiencia temporal en el entrenamiento (Vaswani et al., 2023). Los LLMs se entrenan con **aprendizaje autosupervisado** ( explicado en el apartado 2.1.1.1 ) de tal forma que a partir de una frase, el modelo busca predecir la siguiente palabra. Además, se pueden utilizar técnicas de **fine-tuning** para ajustar al modelo en cuestiones específicas (Radford et al., 2018). Estos modelos se entrenan con masivas cantidades de texto (Minaee et al., 2025), aumentando el poder de generalización.

Particularmente, los LLMs del mercado son decentes generando código. Modelos como GPT-4 y Claude 2 han demostrado ser capaces de resolver problemas fuera de su distribución de entrenamiento, manipular estructuras de datos complejas y ejecutar funciones recursivas con alta precisión (Yang et al., 2025). En este proyecto se utilizan modelos GPT.

#### 2.1.1.5. Modelos Generativos de Imágenes

Por otro lado, también se hace uso de modelos generativos de imágenes, en específico, *gpt-image-1*. Existen varios tipos de modelos generativos pero el más relevante en este caso es el de difusión. El proceso de entrenamiento de estos se basa en agregar ruido, a través de una serie de pasos, a una imagen hasta volverla completamente imperceptible. De esta forma, el modelo aprende a predecir el ruido y a eliminarlo, reconstruyendo así la imagen (Yang et al., 2024).

Modelos como *gpt-image-1* son del tipo **text-to-image**, lo que refiere a la tarea de generar imágenes a partir de un texto descriptivo (Du et al., 2022). De esta forma, se puede especificar al modelo que características debe tener la imagen y este se encarga de generarla.


Figura 3: proceso de entrenamiento y generación de imágenes con modelos de difusión (Yang et al., 2024).

### 2.1.2. Videojuegos

Un concepto importante que se debe tener en cuenta a lo largo de este trabajo son los videojuegos. Desde un punto de vista del diseño, los videojuegos son una actividad de resolución de problemas, pero con un ambiente lúdico (Schell, 2019). Desde el punto de vista técnico, los videojuegos son juegos llevados a cabo por un programa informático. En concreto, la computadora tiene tres roles principales (Jouni Smed y Harri Hakonen, 2003) durante un juego en ejecución:

1. Coordinar el proceso del videojuego.
2. Renderizar / Ilustrar el videojuego.
3. Participar como jugador.

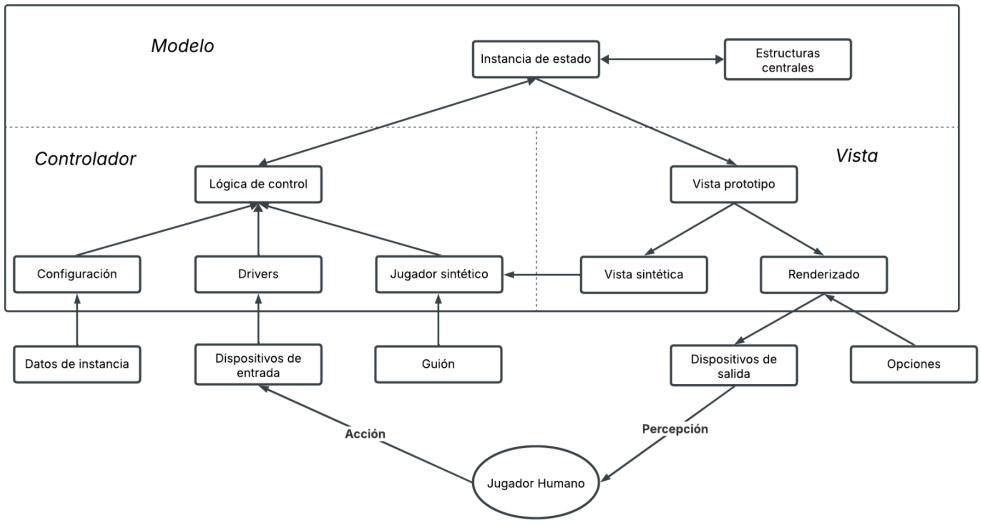


Figura 4: Diseño de un videojuego corriendo en una computadora según el patrón Modelo-Vista-Controlador (MVC) (Jouni Smed y Harri Hakonen, 2003).

### 2.1.2.1. Géneros de videojuegos

Los videojuegos pueden catalogarse en distintos géneros según distintos criterios como hardware y contenidos, entre otros (Adelaida Trujillo Vázquez et al., 2015). Por ejemplo, se cuenta con juegos en dos dimensiones (2D) o tres dimensiones (3D) de Acción, Aventura, Deportes y Plataformas. En particular, este último y en 2D es en el que se centra el proyecto.

Los videojuegos de plataformas son juegos en donde la jugabilidad radica en moverse y saltar entre obstáculos y objetos conocidos como plataformas (Gustafsson, 2014). El primer videojuego que dio popularidad al género fue Space Panic en 1980. Sin embargo, un año después Nintendo inspirado en Space Panic lanzó Donkey Kong para después en 1983 sacar Mario Bros, dos videojuegos de plataformas pero con escenarios fijos. Por último, el videojuego que impulsó

de manera mundial el género fue en 1985, también por Nintendo, Super Mario Bros, ahora sí, un videojuego que avanzaba de manera horizontal (Egenfeldt-Nielsen et al., 2024).

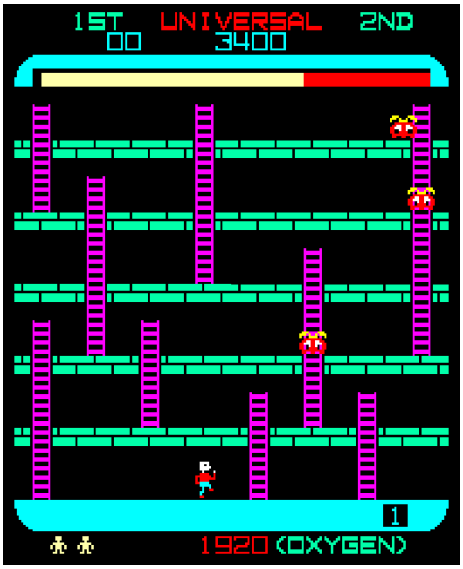


Figura 5: Space Panic (1980) por Universal. (Imagen tomada a través de un emulador online, 2025).

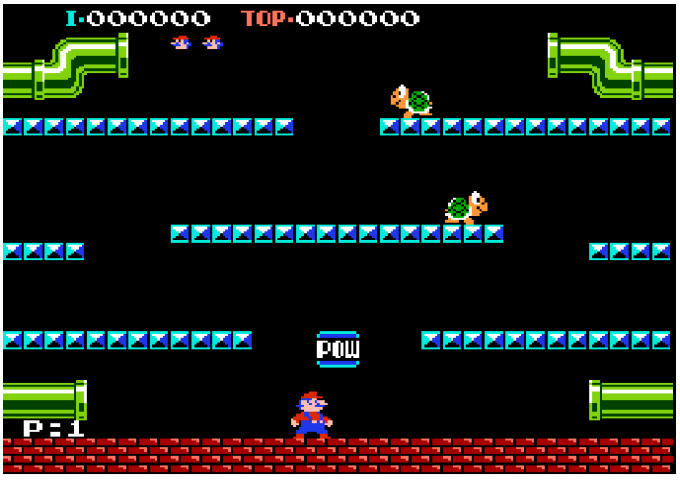


Figura 6: Mario Bros (1981) por Nintendo. (Imagen tomada a través de un emulador online, 2025).

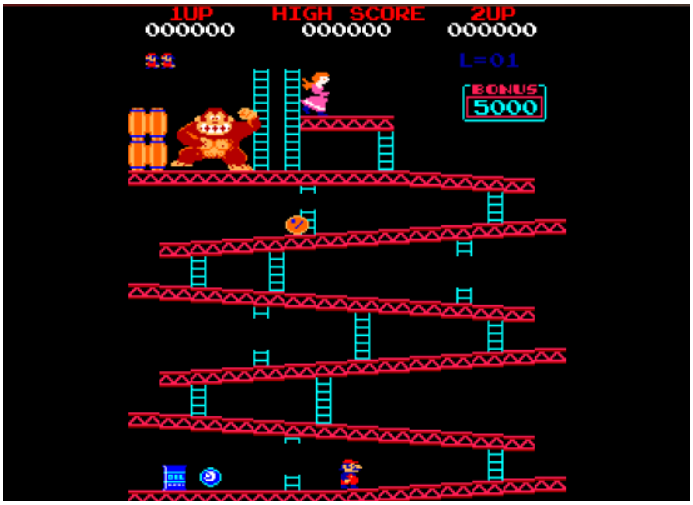


Figura 7: Donkey Kong (1983) por Nintendo.
(Imagen tomada a través de un emulador online, 2025).

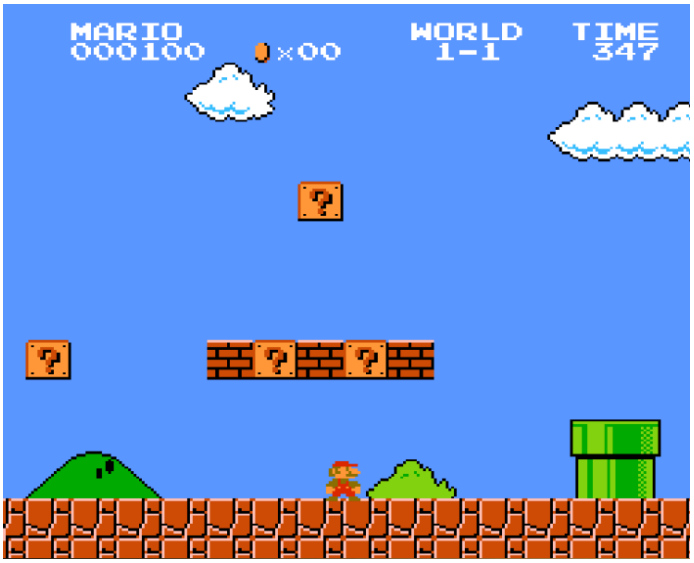


Figura 8: Super Mario Bros (1985) por Nintendo.
(Imagen tomada a través de un emulador online, 2025).

#### 2.1.2.2. Motores de videojuegos

La mayoría de videojuegos modernos se desarrollan utilizando motores de videojuegos. Estos son un conjunto de componentes de software reutilizables que proporcionan funcionalidades esenciales como renderización gráfica, detección de colisiones, etc. y permite a los desarrolladores crear múltiples videojuegos sobre la misma base. En general, los motores de videojuegos están formados por capas en donde la capa de arriba depende de la capa de abajo (Gregory, 2018).

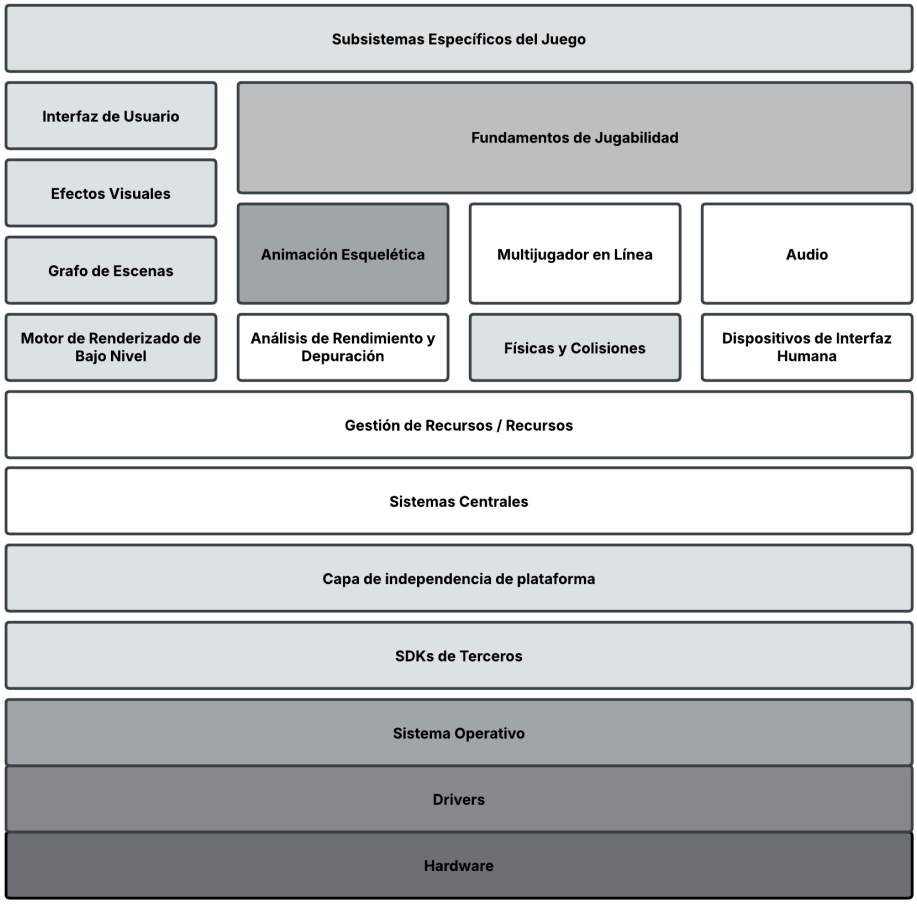


Figura 9: Arquitectura por capas de un motor de videojuegos de propósito general (Gregory, 2018).

### 2.1.2.3. Proceso de desarrollo de un videojuego

Un típico desarrollo de un videojuego consta de tres fases principales: Preproducción, Producción y Post-producción (Aleem et al., 2016). En la primera, se definen los requerimientos, el diseño del videojuego y los prototipos. La producción contempla el desarrollo de activos, programación e implementación haciendo uso de un motor de videojuegos. Finalmente, la última fase se centra en las pruebas, aseguramiento de calidad y estrategias de marketing.

En adición, se puede destacar que la industria de los videojuegos ha adoptado prácticas regulares de la ingeniería de software como las metodologías ágiles. En particular, se puede hablar de cuatro tipos principales de metodologías que utilizan los proyectos de videojuegos: cascada, iterativo, híbrido y ad-hoc. Sin embargo, últimamente, en mayor medida los proyectos se llevan a cabo con un enfoque iterativo (Politowski et al., 2016).

### 2.1.2.4. Prototipado de Videojuegos

Como se menciona en la anterior sección, el prototipado de videojuegos toma relevancia en la etapa de pre-producción. Es esencial para ayudar a los desarrolladores a aclarar cuestiones relacionadas con la jugabilidad y las mecánicas, y además a evaluar la experiencia de juego (Aleem et al., 2016). Y, justamente, el prototipado de videojuegos cumple un rol fundamental que va más allá de la validación técnica. Permite materializar ideas abstractas, evaluar mecánicas específicas y comunicar intenciones de diseño entre miembros del equipo. Entonces, se transforman ideas en objetos interactivos que pueden ser evaluados, ajustados y posteriormente incorporados al diseño final (Manker y Arvola, 2011). Este enfoque iterativo centrado en la experiencia del jugador diferencia el prototipado de videojuegos del de otras industrias del software.

Sin embargo, en ocasiones, el prototipado se vuelve prolongado. En estos casos, pueden surgir problemas como falta de dirección, toma de decisiones pospuestas, sesgos en la evaluación de ideas y una sensación falsa de progreso. En lugar de acelerar el desarrollo, puede convertirse en una carga productiva constante, con costos elevados y escasos beneficios si no está correctamente acotado y orientado (Margaris, 2025).

## 2.2. Estado del Arte

En los últimos años, han surgido diversas herramientas relacionadas con inteligencia artificial y el desarrollo de videojuegos. A continuación, se exploran servicios de motores de propósito general, motores orientados al prototipado y proyectos académicos sobre la inteligencia artificial aplicada al desarrollo de videojuegos.

### 2.2.1. Motores de propósito general

#### 2.2.1.1. Unity Muse, servicio de Unity

Unity es un motor de videojuegos comercial y de propósito general muy utilizado en la industria por empresas desarrolladoras de videojuegos así como también equipos independientes. Es muy utilizado para desarrollar videojuegos en 3D y 2D de alta calidad y ofrece soporte para múltiples plataformas, tanto computadoras como consolas y celulares («Real-Time 3D Development Platform & Editor», [sin fecha]). En particular, Unity Technologies ofrece una herramienta de inteligencia artificial llamada Unity Muse, un chat integrado al motor que permite generar texturas, animaciones, código y explorar distintas soluciones. Esto es ofrecido a 30 dólares por cada empleado que lo use dentro de la misma licencia del motor («Unity Muse», [sin fecha]). Sin embargo, muchos usuarios han expresado su descontento con el sistema de suscripción ya que ofrece “puntos” mensuales que van disminuyendo con cada acción en la herramienta. Además, no existe mucha información relacionada a cómo se consumen estos puntos por lo que la empresa no es muy transparente al respecto. («How does everyone actually feel about the points system for Muse?», 2025; «Unity muse price disappointment - Muse», 2023).

Este motor de videojuegos, al ser utilizado para desarrollar distintos tipos de videojuegos, ofrece herramientas y servicios como las que se observan en la figura 9 y por esta razón, no está centrado en prototipado, sino en ofrecer un producto de uso general.

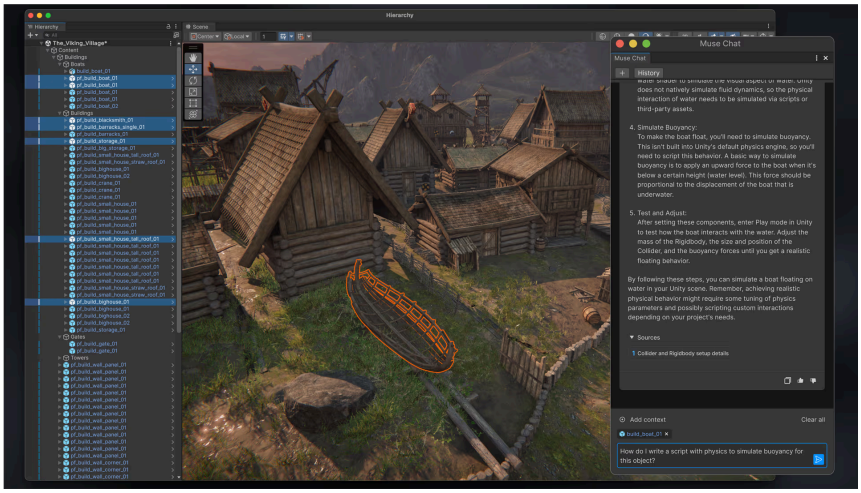

Figura 10: Unity Muse, se observa un chat integrado al motor a través del cual explorar soluciones («Unity Muse», [sin fecha]).

### 2.2.1.2. AI Assistant Hub, Extensión de Godot Engine

Godot es un motor de videojuegos de código abierto, gratis y manejado por la comunidad que permite crear videojuegos 2D y 3D. Con él, se puede realizar cualquier tipo de videojuego y para cualquier plataforma sin restricción. Una de las características del mismo es la facilidad de uso («Introduction», [sin fecha]).

El motor cuenta con Asset Library, una plataforma a través de la cual los distintos desarrolladores pueden descargar extensiones o activos, elaborados por la comunidad, para su propio proyecto de forma gratuita. Algunas de estas extensiones ofrecen soporte para herramientas de inteligencia artificial. En particular, “AI Assistant Hub” es una extensión que permite generar código y explorar distintas soluciones al comunicarse con modelos de inteligencia artificial de manera local o remota a cargo del usuario («AI Assistant Hub - Godot Asset Library», [sin fecha]). Si bien este activo ayuda a optimizar el proceso, en términos de generación, solo se encarga del código y no de texturas o sprites.

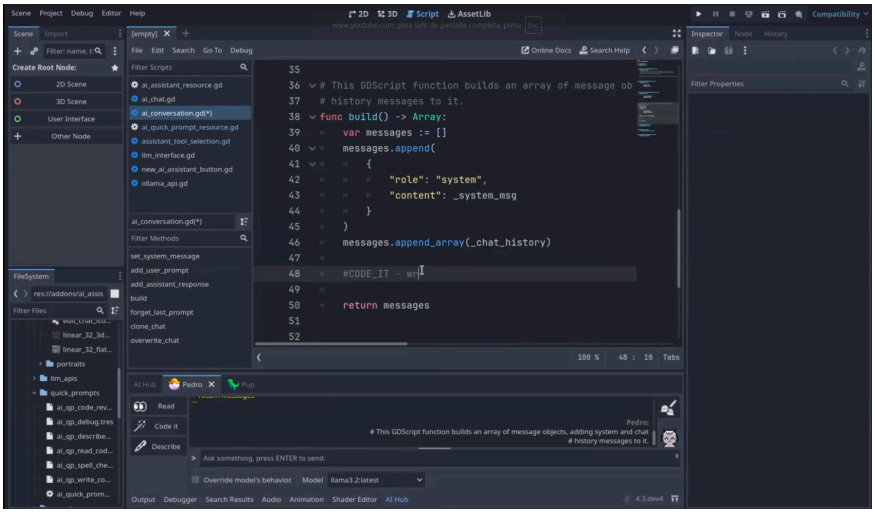


Figura 11: AI Assistant Hub, extensión de godot engine integrada a la interfaz del motor («Godot AI Assistant Hub (1/6) - What is it?», 2024).

## 2.2.2. Motores y plataformas orientadas al prototipado

### 2.2.2.1. Ludo

Ludo es una plataforma orientada a asistir en la etapa de pre-producción de un videojuego. Utiliza tecnologías de inteligencia artificial para ofrecer un conjunto de funcionalidades que integran generación de texto, generación de imágenes, creación de modelos 3D, generación de código, y herramientas de búsqueda e inteligencia de mercado potenciadas por IA. Cabe destacar que Ludo.ai no permite crear videojuegos completos directamente. Su propósito principal es acelerar la primera etapa del proceso de desarrollo de un videojuego. («Create hit games with the power of AI with Ludo», [sin fecha]).

Una de las funcionalidades que ofrece es la capacidad de generar experiencias jugables a partir de una descripción o prompt. Al empezar a utilizarla se pide al usuario seleccionar si quiere un motor 2D con PixiJS o 3D con Three.js, dos librerías gráficas en lenguaje de programación javascript. Sin embargo, dado que se trata de una capacidad recientemente incorporada, durante las pruebas se observaron fallos de ejecución (figura 12). Además, permite cargar imágenes propias para sumar a la experiencia y también revisar el código generado aunque no es posible editarlo.

Para la gestión de errores, la interfaz incluye el control **“Fix error”**, que captura el mensaje de la consola y lo remite al modelo de IA a través del chat para su reintento de corrección**.** En el escenario evaluado, tras dos iteraciones el modelo resolvió la excepción y la aplicación presentó un videojuego de plataformas básico (Figura 13). Sin embargo, el parámetro de fuerza de salto resultó insuficiente, por lo que la meta(estrella) no era alcanzable. En una última iteración, se le pidió al modelo mediante el chat que aumente la fuerza de salto y genere una textura de pasto para el suelo, de piedra para las plataformas y que convierta al personaje en un cerdo, aunque como se visualiza en la figura 14, el resultado fue simplemente un cambio en el color de los mismos.

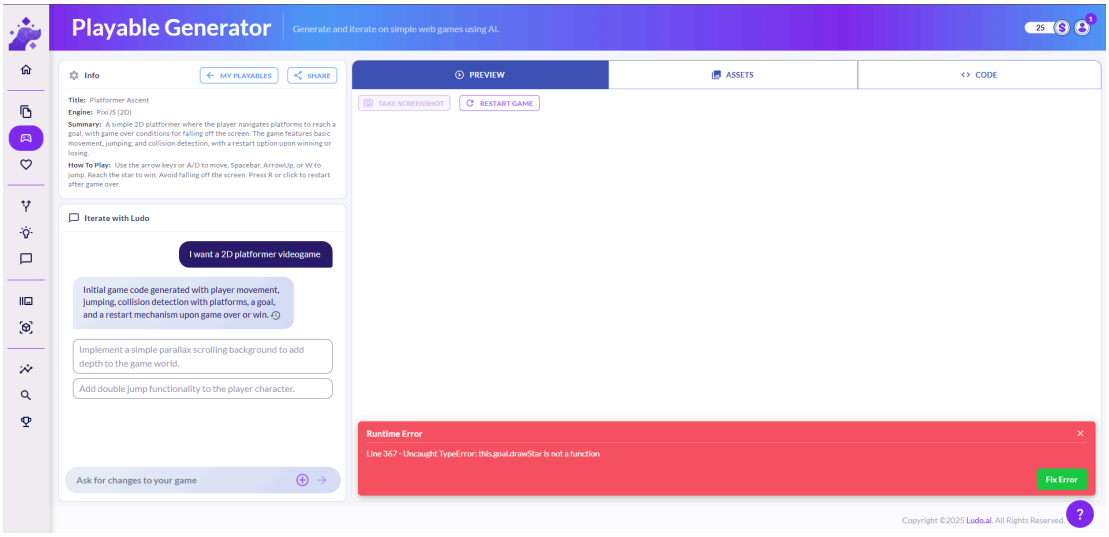


Figura 12: generador jugable de Ludo. Se observa un error al intentar generar un juego 2D de plataformas. (Imagen capturada de la plataforma Ludo.AI, 2025).

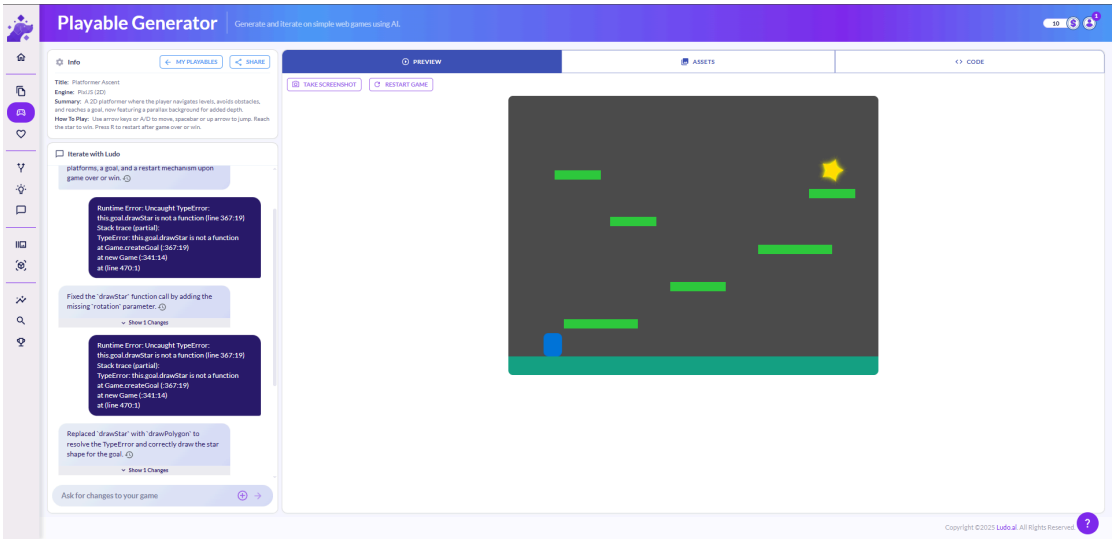


Figura 13: generador jugable de Ludo. Experiencia jugable sin posibilidad de victoria. (Imagen capturada de la plataforma Ludo.AI, 2025).

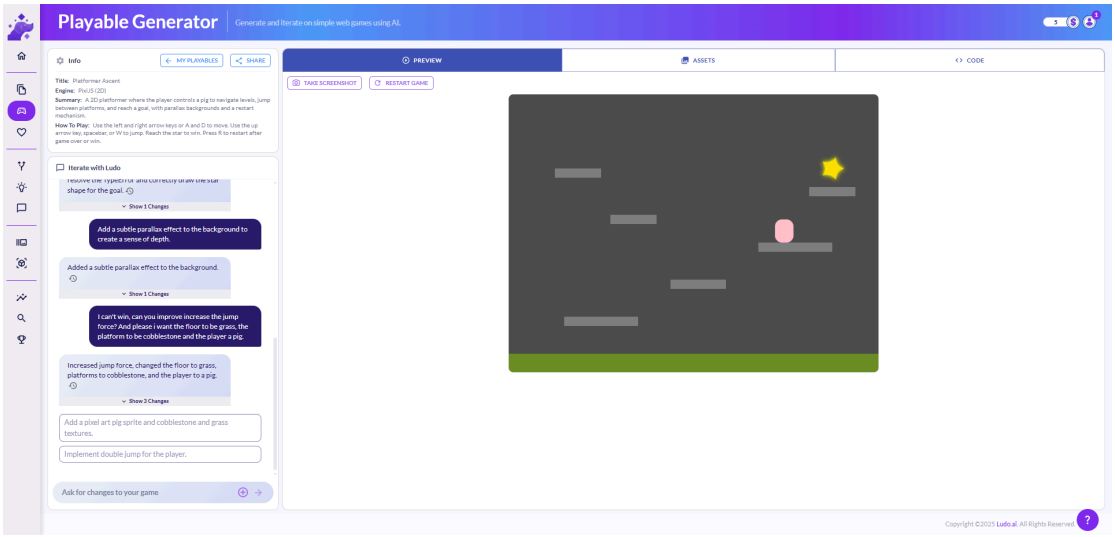


Figura 14: generador jugable de Ludo. Aumento en la potencia de salto y cambio de colores de los recursos. (Imagen capturada de la plataforma Ludo.AI, 2025).

En cuanto al modelo de negocio, se ofrece una versión gratuita limitada a 30 créditos y tres modelos distintos de suscripción pensados para distintos niveles de poder adquisitivo y tamaño de equipo. Estos son:

- Indie (15 dólares mensuales):
  - Una licencia.
  - Hasta cinco proyectos.
  - 500 créditos mensuales.
- Pro (30 dólares mensuales):
  - Una Licencia.
  - Proyectos ilimitados.
  - Créditos ilimitados para ideación, desarrollo y generación de recursos.
  - 1000 créditos por mes para generaciones jugables.
  - Soporte premium.
  - Integración con terceros.
- Studio (80 dólares mensuales):
  - 10 licencias
  - Proyectos ilimitados.
  - Créditos ilimitados para ideación, desarrollo y generación de recursos.
  - 1000 créditos por mes para generaciones jugables.
  - Soporte premium.
  - Colaboración en tiempo real.
  - Integración con terceros.

La mayoría de funcionalidades del producto cuestan 1 crédito exceptuando la herramienta para generar experiencias jugables que vale 5 créditos por acción generada, incluso si esta se trata de arreglar un error.

## 2.2.3. Proyectos académicos

### 2.2.3.1. MarioGPT

A través de un proyecto académico, se desarrolló MarioGPT es un modelo basado en GPT-2 que ha sido entrenado para generar niveles de Super Mario Bros, se dice que es un enfoque "text-to-level". Mediante aprendizaje autosupervisado, el modelo fue entrenado utilizando representaciones de los niveles del juego en formato de cadenas de texto en donde cada símbolo equivale a un objeto del juego. El modelo es capaz de generar niveles jugables y coherentes a través del lenguaje natural permitiendo explorar aspectos del diseño de niveles como la dificultad o las estéticas. A pesar que puede generar diversos contenidos, MarioGPT tiene sus limitaciones ya que a veces no se ajusta a los prompts y está sujeto a memorizar el dataset, es decir, pierde la capacidad de generalizar (Sudhakaran et al., 2023).


Figura 15: equivalencia entre un nivel en formato texto y el nivel dentro del juego (Sudhakaran et al., 2023).

#### 2.2.3.2. Uso de inteligencia artificial generativa en el desarrollo de videojuegos

Diversos estudios han denotado ciertas ventajas y desventajas respecto a la utilización de la inteligencia artificial generativa en el proceso de desarrollo de videojuegos (Andrew Begemann y James Hutson, 2024; Alharthi, 2025).

Se perciben las siguientes ventajas:

- Aumento de productividad: influye en la aceleración del proceso, especialmente en etapas tempranas del desarrollo.
- Reducción de la brecha técnica: al tratarse con lenguaje natural, permite el involucramiento de las personas que no tienen conocimientos técnicos.
- Exploración de ideas: debido a la capacidad de generar múltiples versiones de los recursos, facilita la búsqueda de conceptos de juego.

Por otro lado, las próximas desventajas son identificadas:

- Falta de control sobre resultados: la falta de precisión en las generaciones puede ser frustrante.
- Problemas éticos y de propiedad intelectual: la falta de regulación y transparencia emergen debates controversiales.
- Sobre dependencia: es posible que las personas desarrollen una dependencia, perdiendo habilidades creativas en el transcurso del tiempo.

### 2.2.4. Tabla comparativa entre los productos y herramientas

A continuación, se presenta una tabla comparativa de los productos y herramientas descritas. Quedan fuera de la tabla los proyectos académicos ya que no tienen la intención de ser un producto formal, están centrados en explorar soluciones o generar nuevos conocimientos.

Tabla I: Comparación de productos con Game Protogen. (Elaboración propia, 2025).

| | **Game Protogen** | Unity Muse | AI Assistant Hub | Ludo.AI |
|---|---|---|---|---|
| Enfocado en prototipos | **Si** | No | No | Si |
| Motor de videojuegos | **Si** | Si | Si | No |
| Producto integral concebido con inteligencia artificial | **Si** | No | No | Si |

### 2.2.5. Conclusión

En conclusión, se puede notar una creciente tendencia en el uso de la inteligencia artificial generativa y su integración en el proceso de desarrollo de un videojuego. Existen productos y extensiones, como Unity Muse y las extensiones de Godot que están presentes en el desarrollo completo del videojuego; y plataformas como Ludo que están pensadas para agilizar la etapa de pre-producción. Los productos comerciales usan en su mayoría modelos de negocio basados en suscripción.

Por otro lado, además hay que tener en cuenta el surgimiento de varios estudios y proyectos en relación con esta temática que por un lado enfatizan el uso de la inteligencia artificial generativa como una herramienta que optimiza el proceso de desarrollo, pero por el otro denotan cuestiones vinculadas con la ética, la moral y la dependencia en la misma.

# 3. Descripción

## 3.1. User Research

Con la finalidad de validar la problemática que este proyecto busca resolver se realizó una investigación mediante encuestas y entrevistas con el propósito de obtener datos primarios. A continuación, se presenta un análisis de los resultados.

### 3.1.1. Encuestas

Para obtener una visión general de cómo se sienten los desarrolladores de videojuegos frente a la etapa de prototipado, se desarrolló una encuesta que fue distribuida a lo largo de diversas comunidades de desarrolladores de videojuegos obteniendo finalmente 72 respuestas. La cantidad de respuestas se considera suficiente ya que el tamaño de la muestra es representativo del tamaño de la industria de los videojuegos en el país (Rossi et al., 2024).

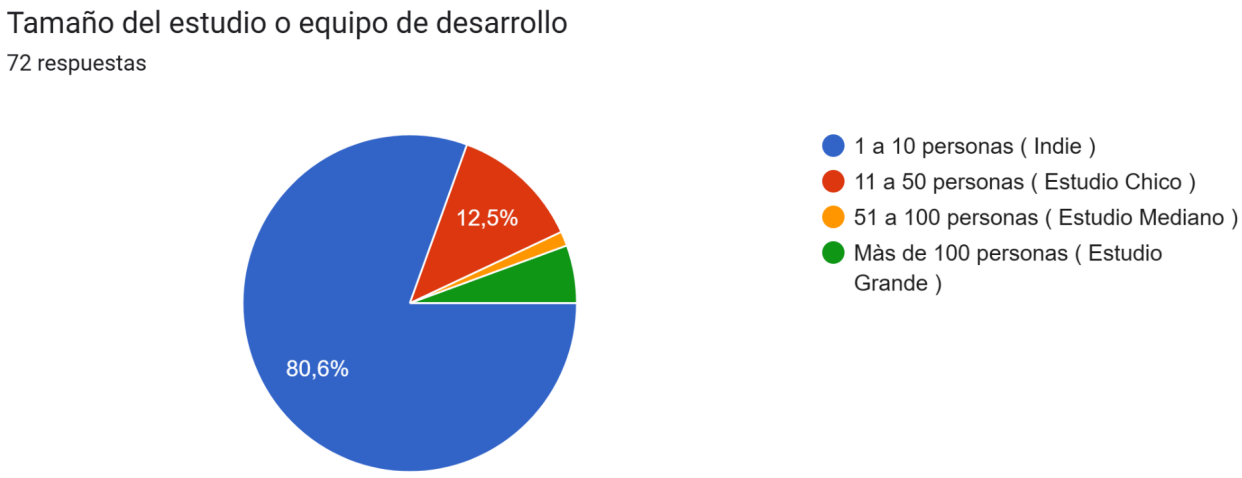


Figura 16: respuestas de encuesta en referencia al tamaño del estudio o equipo del encuestado. (Gráfico de elaboración propia, 2025).

Para empezar, el **80%** de los encuestados expresaron que su tamaño de equipo era de 1 a 10 personas, es decir, equipos indie. Este dato es importante ya que permite pensar en que el usuario target del producto son estudios indie, y, relacionado con esto, se sabe que estos tipos de equipos en la industria del desarrollo de videojuegos están envueltos en un ambiente de experimentación continua para explorar, diseñar y desarrollar nuevas ideas de juego y características (Linåker et al., 2024).


Figura 17: respuestas de encuesta en referencia al nivel de acuerdo o desacuerdo con la afirmación “La etapa de prototipado influye significativamente a la hora de desarrollar un videojuego”. (Gráfico de elaboración propia, 2025).

Para despejar dudas, se preguntó acerca de la importancia de la etapa de prototipado a la hora de desarrollar un videojuego, y se obtuvo que más del **90%** de los encuestados están de acuerdo o muy de acuerdo con que la etapa de prototipado influye de manera significativa en el proceso de desarrollo de un videojuego. Esto sugiere que los desarrolladores están abiertos a reflexionar sobre esta etapa ya que existe una conciencia compartida sobre su valor.

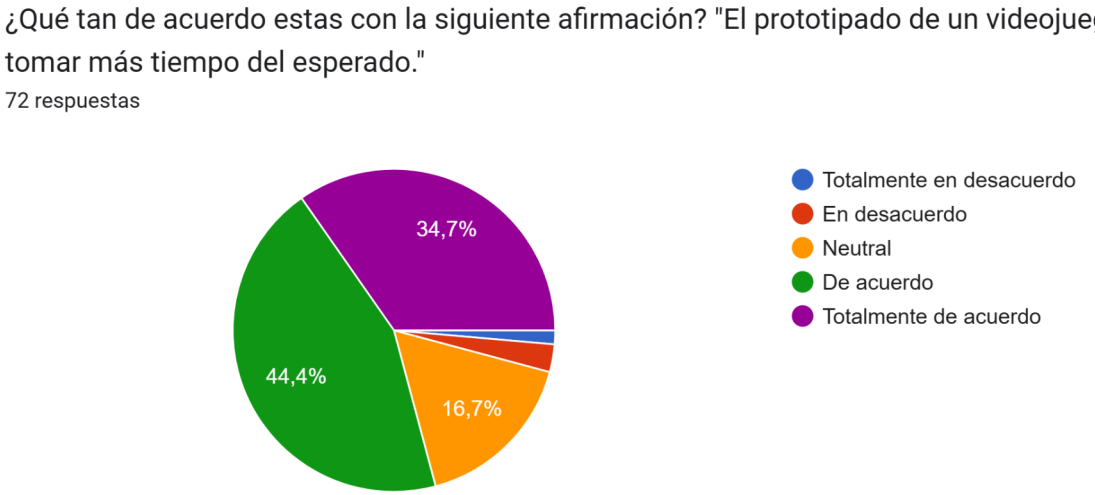


Figura 18: respuestas de encuesta en referencia al nivel de acuerdo o desacuerdo con la afirmación “El prototipado de un videojuego puede tomar más tiempo del esperado”. (Gráfico de elaboración propia, 2025).

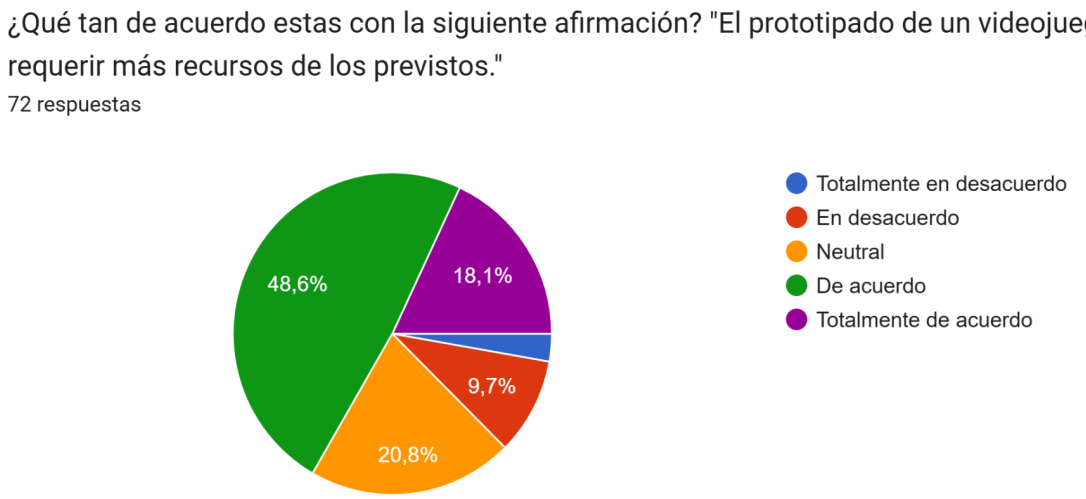


Figura 19: respuestas de encuesta en referencia al nivel de acuerdo o desacuerdo con la afirmación “El prototipado de un videojuego puede requerir más recursos de los previstos”. (Gráfico de elaboración propia, 2025).

Para validar la problemática de manera directa, se consultó el grado de de acuerdo o desacuerdo respecto a que el prototipado de videojuegos toma más tiempo del esperado y más recursos de los previstos. Respecto a lo primero, más del **75%** de los encuestados estaban de acuerdo o totalmente de acuerdo, y en relación con lo segundo, más del **65%**. Esto indica que la problemática que dió pie a este proyecto no es un mero pensamiento del autor del mismo, sino que más de la mitad de los encuestados coincidieron con el planteo.

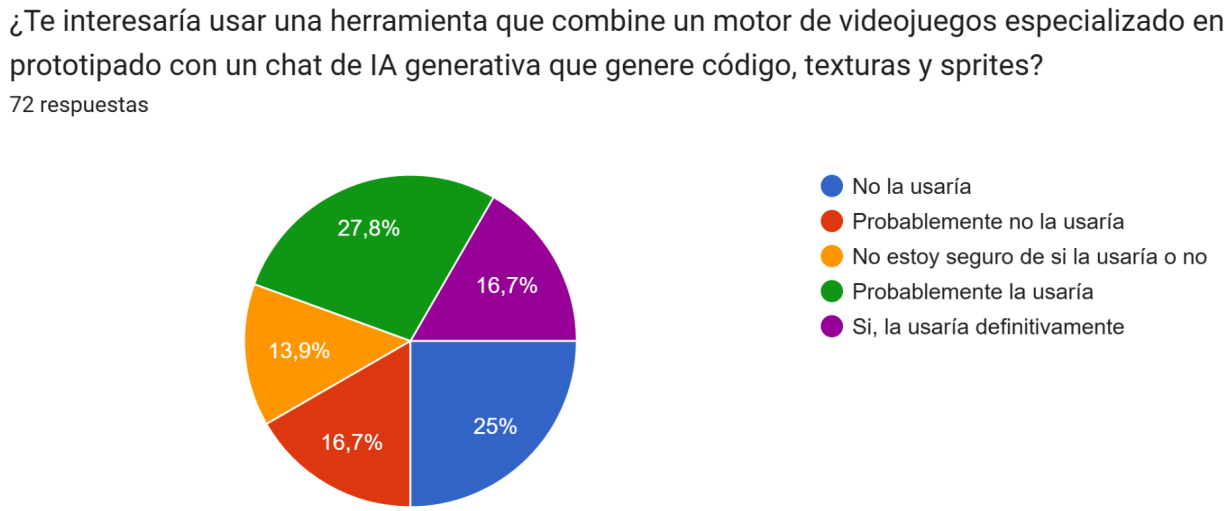


Figura 20: respuestas de encuesta en referencia a si los encuestados usarían una herramienta como la que plantea este proyecto. (Gráfico de elaboración propia, 2025).

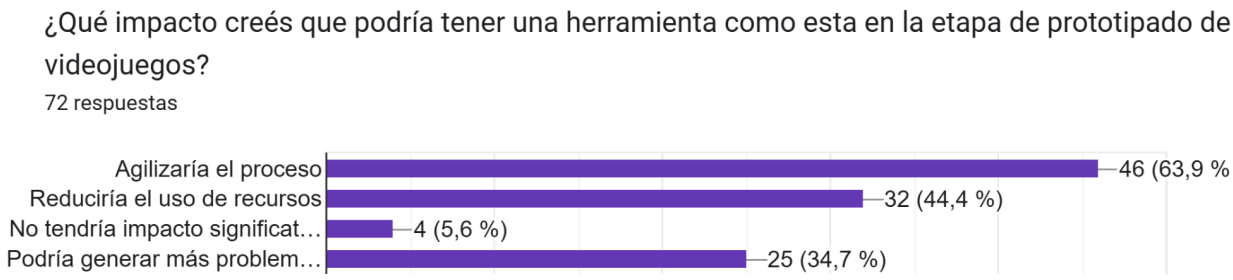


Figura 21: respuestas relevantes de la encuesta en referencia al impacto que la herramienta planteada puede tener para los encuestados en la etapa de prototipado. (Gráfico de elaboración propia, 2025).

Para finalizar, se preguntó de manera directa sobre si los encuestados utilizarían la herramienta que sugiere este proyecto y el impacto que ellos creen que la misma puede tener en la etapa de prototipado. Principalmente, se observa que el **44,5%** de los encuestados expresaron que usarían la herramienta, y un **13,9%** no está seguro de si la usarían o no. En contraste, un **41,7%** de los encuestados eligieron opciones negativas frente al uso de la herramienta.

Respecto al impacto de la herramienta, todos aquellos que coincidieron en el uso de la herramienta también señalaron que esta agilizaría el proceso, reduciría el uso de recursos, o ambas cosas. De los 10 encuestados que no estaban seguros sobre el uso de la herramienta, 9 eligió las opciones positivas. Por último, el **46,6%** de los encuestados que no usarían el producto de manera probable o definitiva, igualmente marcaron opciones positivas en relación con el impacto.

### 3.1.2. Entrevistas

Se entrevistó a dos personas relacionadas de manera directa con la industria y el desarrollo de los videojuegos en Argentina.

La primera entrevista se realizó a Juan Pablo Cardoso, desarrollador de videojuegos independiente. Actualmente está desarrollando su videojuego **Mind The Clown**. Ha trabajado en distintas empresas de videojuegos y además ha desarrollado diversos videojuegos. El objetivo de esta entrevista fue entender el proceso de prototipado para un desarrollador, validar la problemática de manera directa con él, preguntar sobre su opinión frente al uso de la inteligencia artificial y consultarle si usaría la herramienta propuesta en este trabajo.

Por otro lado, la persona entrevistada en segundo lugar fue Cristian Basoalto, director de **Bacord Games**, un estudio indie argentino. La finalidad de la entrevista en este caso se centró en comprender la visión del prototipado en un estudio, el tiempo y recursos dedicados a la etapa. Además, también se exploró la perspectiva de la inteligencia artificial y el posible uso del producto en su estudio.

#### 3.1.2.1. Juan Pablo Cardoso ( Desarrollador de videojuegos indie )

Juan, a la hora de prototipar, empieza buscando inspiración en alguna temática que le guste. En base a eso, desarrolla las mecánicas. Él crea una lista de mecánicas primarias y desarrolla el prototipo desde lo más principal hasta lo más secundario resultando en una versión base que tiene el objetivo de evaluar si la jugabilidad básica del juego es divertida y llama la atención. Si no logra un prototipo con el que se sienta conforme, a través de la iteración lo ajusta para llegar al punto deseado.

Respecto a la desviación del tiempo planificado para realizar el prototipo con el tiempo que le lleva en realidad, Juan expresó que el prototipo del juego actual que está desarrollando se extendió más de lo que él había previsto. Además hablaba de que el ida y vuelta entre ir a buscar o crear recursos para el juego lo sacaba del “flow” del desarrollo, ralentizandolo.

En relación con la inteligencia artificial, Juan comentó que si bien hay desafíos, también existen oportunidades. Para él, esta tecnología es una herramienta que agiliza el proceso. Sin embargo, a la hora de generar imágenes, texturas, sprites, hay que tener ciertas cuestiones a considerar como la resolución.

Por último, Juan opina que hay retos en el desarrollo de una herramienta como la que plantea este proyecto pero piensa que serviría a la hora de prototipar un videojuego.

#### 3.1.2.2. Cristian Basoalto ( Director de Bacord Games )

Para empezar, Cristian menciona que el prototipado de videojuegos es crucial. Para él, el prototipado está subestimado. Si funciona bien el prototipo, funcionará bien en el mercado. Es por eso que en estudios independientes como Bacord Games donde se tienen recursos y tiempos limitados para dedicar de lleno a esta etapa él siente que un producto enfocado en la misma sería clave. Además, Cristian plantea que una herramienta de este estilo reduciría la brecha entre lo técnico y creativo, permitiendo que alguien que no es programador pueda crear un prototipo e iterar centrándose, justamente, en el lado del diseño.

Acerca de la desviación entre tiempos y recursos planificados y reales, Cristian expresó que en su empresa, de manera análoga, tienen un cajón de cocina lleno de prototipos que por falta de tiempo y recursos no pudieron terminar de consolidar.

Para terminar, en cuanto a la inteligencia artificial destaca que si bien, faltan regulaciones y existen debates morales y éticos sobre su uso, es una herramienta que ayuda en el desarrollo.

### 3.1.3. Observaciones

En conclusión, el user research refleja que la problemática planteada es un hecho en la práctica del desarrollo de videojuegos. Tanto los entrevistados como más de la mitad de los encuestados señalaron que la planificación del prototipado de videojuegos suele tener diferencias con la realidad en términos de tiempo y recursos. De esta forma, el grupo consultado se mostró abierto e interesado en una herramienta como la que plantea este proyecto, enfatizando en que puede optimizar el flujo de trabajo y la etapa de prototipado en particular.

## 3.2. Análisis funcional

Para entender las capacidades que la aplicación ofrece y garantiza, primero se describen los casos de uso y luego, los requisitos de la misma.

### 3.2.1. Casos de uso

A continuación, se exponen los principales casos de uso de **Game Protogen**:

1. **Generar prototipos jugables:** El usuario describe una mecánica o idea en lenguaje natural y la IA genera un prototipo jugable dentro del motor con sprites, texturas y lógica básica.
2. **Iterar sobre un prototipo generado:** El usuario solicita cambios y la IA actualiza dinámicamente el prototipo sin necesidad de modificar código manualmente.
3. **Exportar prototipo jugable:** El usuario puede exportar el prototipo para jugarlo por fuera de la aplicación.
4. **Generar niveles a partir de prompts:** El usuario puede generar el layout básico de un nivel a través del chat con IA generativa.
5. **Editar con herramientas básicas visuales:** El usuario puede mover, transformar, editar y transformar las entidades presentes en el prototipo.

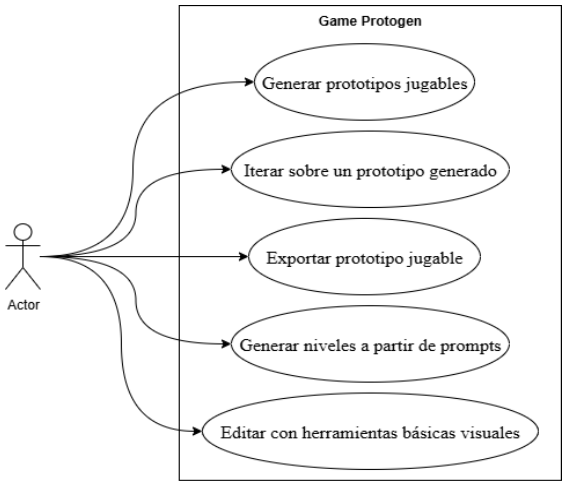


Figura 22: Diagrama de casos de uso.
(Gráfico de elaboración propia, 2025).

### 3.2.2. Requerimientos

Game Protogen cumple con ciertas características. A través de los requisitos funcionales se describe el **QUÉ** y a través de los requisitos no funcionales el **CÓMO** funciona el sistema.

#### 3.2.2.1. Requerimientos Funcionales

El motor **DEBE**:

- Permitir editar la escena en tiempo real.
- Permitir mover, escalar, rotar y cambiar el color de las entidades.
- Permitir probar el prototipo dentro del motor.
- Permitir utilizar scripts para agregar lógica al prototipo a través de métodos expuestos por el motor.
- Cambia textura o sprite de un elemento
- Permitir exportar el prototipo como ejecutable jugable.
- Proveer un chat para comunicarse con la IA.
- Generar prototipos jugables a partir de la conversación con la IA.
- Guardar y cargar la escena.

#### 3.2.2.2. Requerimientos No Funcionales

**COMO** motor de prototipado, debe funcionar garantizando:

- **Usabilidad:**
  - Debe ser intuitivo y sencillo de usar, ofreciendo una interfaz similar a otros motores del mercado, al igual que *shortcuts* para aquellos que estén acostumbrados a utilizarlos.
- **Fiabilidad:**
  - Debe preguntar por el guardado de la escena previo a salir de la aplicación.
  - Debe tolerar los fallos que puedan surgir a través de la comunicación con la IA.

- **Mantenibilidad:**
    - Debe estar diseñado con arquitectura modular que permita agregar nuevas funcionalidades, manteniendo desacoplado el editor visual del motor.

## 3.3. Solución

Game Protogen es un motor de videojuegos especializado en el prototipado y enfocado en videojuegos 2D de plataformas. Su propuesta de valor se centra en 3 elementos:

- **Especialización en el prototipado:** a diferencia de los motores de videojuegos existentes en el mercado que son de propósito general, Game Protogen se centra en el proceso de prototipar un videojuego.
- **Integración con Inteligencia Artificial:** en contraste con otros motores usados en la industria que se sumaron a la tendencia de utilizarla, la IA es la base de Game Protogen.
- **Chat con IA Generativa:** la aplicación hace foco en el chat con el que el usuario puede comunicarse. A través del mismo, se reduce la brecha técnica permitiendo que un artista o diseñador pueda prototipar sin saber programar.

### 3.3.1. Tecnologías y servicios

A lo largo del desarrollo del proyecto, se utilizaron distintas tecnologías que hacen posible el éxito del mismo. Las mismas se describen a continuación.

#### 3.3.1.1. C++

C++ es un lenguaje de programación de propósito general, de alto rendimiento, orientado a objetos y con gran control sobre memoria y recursos. Es ampliamente reconocido como el lenguaje más utilizado en el desarrollo de motores de videojuegos como Unreal Engine, Unity y Godot debido a su eficiencia, cercanía al hardware y soporte para programación en tiempo real.

En este proyecto, el lenguaje se utiliza, justamente, para el desarrollo del motor en conjunto con la librería gráfica **SFML** para renderizado y manejo de entradas. Además, se utiliza **ImGUI** como librería que ofrece sustento en la interfaz gráfica. Su elección responde tanto a la

necesidad de eficiencia y rendimiento como al alineamiento con las prácticas estándar de la industria del desarrollo de motores.

#### 3.3.1.2. .NET

.NET es un framework de desarrollo multiplataforma de Microsoft que permite crear aplicaciones de escritorio, web y servicios backend. En este proyecto se utiliza **.NET 8** para la implementación de la API y la capa de orquestación de la IA generativa, facilitando la conexión entre el motor y los servicios de inteligencia artificial. Proporciona robustez, escalabilidad y facilidad de integración con múltiples librerías y SDKs.

.NET se destaca hoy por su rendimiento optimizado y su fluida integración con entornos Windows y servicios cloud. Al ser gestionado por Microsoft, que en los últimos años ha estado apostando por la inteligencia artificial, se vuelve muy compatible con este proyecto a través de la utilización de librerías que facilitan el manejo de inteligencia artificial que la misma empresa ha liberado.

#### 3.3.1.3. Lua

Lua es un lenguaje de scripting ligero, embebible y de sintaxis simple, diseñado para integrarse dentro de aplicaciones escritas en C/C++. Es ampliamente usado en videojuegos y motores por su curva de aprendizaje baja, tamaño reducido y excelente interoperabilidad con C.

En este proyecto, Lua se utiliza para scripting de gameplay y lógica de entidades sobre el motor. El mismo expone una serie de métodos que pueden ser utilizados por los scripts en lua para interactuar con él. Esto permite que se pueda agregar lógica sin la necesidad de recompilar el motor.

#### 3.3.1.4. Microsoft Azure

Microsoft Azure es una plataforma de nube líder a nivel global. Ofrece servicios IaaS, PaaS y SaaS con alta escalabilidad, disponibilidad y un sólido enfoque en seguridad y cumplimiento. En el proyecto se emplea para el despliegue en la nube de la infraestructura. Azure aporta confiabilidad, flexibilidad para escalar según la demanda del motor e integración nativa con el ecosistema de Microsoft (Windows, Active Directory y analítica, entre otros).

En particular, Azure ofrece un servicio llamado Azure Foundry. Este es un ecosistema de inteligencia artificial a través del cual se puede acceder a distintos modelos desde un solo lugar y conectarlos a través del mismo proveedor de nube.

#### 3.3.1.5. OpenAI

OpenAI es una organización de investigación y desarrollo de inteligencia artificial, creadora de modelos.

En particular, en el presente proyecto se utiliza gpt-image-1 para generar imágenes en alta calidad e incluso con fondos transparentes.

#### 3.3.1.6. Git

Git es un sistema de control de versiones distribuido. Se ha convertido en el estándar de la industria para el versionado de código gracias a su velocidad, flexibilidad y capacidad de trabajar en entornos distribuidos.

En este proyecto, Git asegura que el desarrollo del motor y de la API pueda mantenerse organizado, trazable y recuperable.

#### 3.3.1.7. Github

GitHub es una plataforma en la nube que extiende a Git, proporcionando un entorno colaborativo que permite a equipos de desarrollo trabajar de manera integrada y remota.

En este proyecto, GitHub se utiliza para centralizar el repositorio del motor y la API. Además, esta plataforma brinda soporte a un tablero KANBAN a través de Github Projects y permite acciones de despliegue continuo e integración continua a mediante GitHub actions.

### 3.3.2. Arquitectura de alto nivel

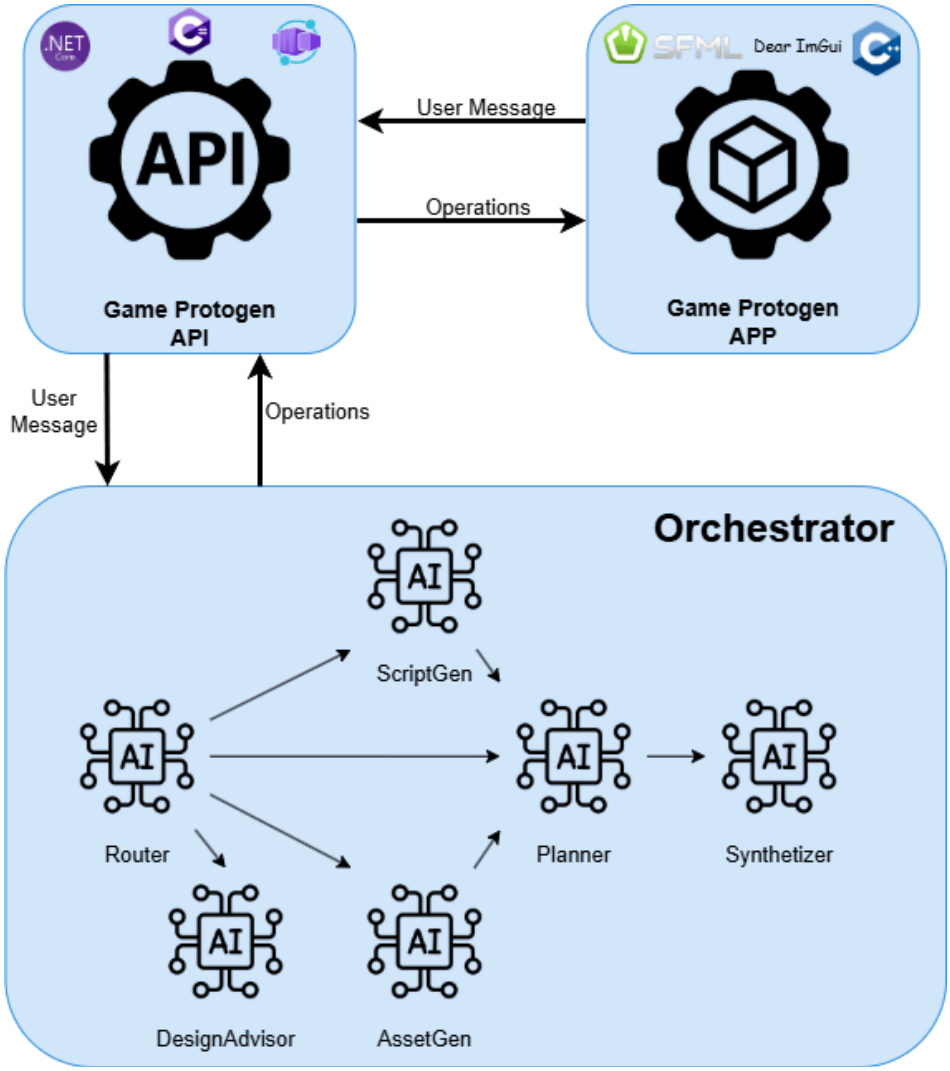


Figura 23: Arquitectura de alto nivel de Game Protogen. (Gráfico de elaboración propia, 2025).

La arquitectura de Game Protogen está basada alrededor de 4 elementos principales:

- **Game Protogen App:** El motor en sí. Actúa como cliente de la API. Es a través de lo que el usuario interactúa generando mensajes para la IA o editando el prototipo manualmente.
- **Game Protogen API:** Se encarga de mediar la comunicación entre el orquestador de agentes y la aplicación. Es el servidor al que el cliente envía peticiones.

- **Orquestador de Agentes:** Recibe el mensaje del usuario por medio de la API y comienza un proceso secuencial con el objetivo de generar una estructura de datos que el motor entienda. De esta forma, el resultado de ese proceso desemboca en un protocolo que la app puede procesar para generar lo pedido por el usuario.

Por lo tanto, el flujo de datos es el siguiente:

1. El usuario envía un mensaje desde la APP, por ejemplo: “Quiero generar 5 plataformas azules”.
2. El mensaje llega a la API, que lo reenvía al orquestador.
3. Los agentes IA procesan el mensaje de forma secuencial.
4. El resultado (operaciones) vuelve a la API y de ahí al motor APP, que aplica los cambios al prototipo, generando las 5 plataformas solicitadas.

Respecto a los agentes, se identifican los siguientes:

- **Router:** este agente recibe el mensaje del usuario y devuelve como resultado la lista de agentes a los que hay que llamar.
- **Design Advisor:** responde preguntas de diseño de videojuegos.
- **Asset Gen:** genera texturas y sprites para las entidades.
- **Script Gen:** crea código para aplicar lógica al prototipo.
- **Planner:** obtiene todo lo generado y hace un plan para modificar la escena.
- **Synthesizer:** traduce lo planificado a un protocolo que el motor entienda.

### 3.3.3. Arquitectura del motor

Siguiendo la arquitectura de un motor de propósito general descrita a través de la figura 9, se puede establecer una figura análoga, aunque con menos capas, que representa al motor Game Protogen. A continuación se procede a explicar los distintos elementos que la componen:

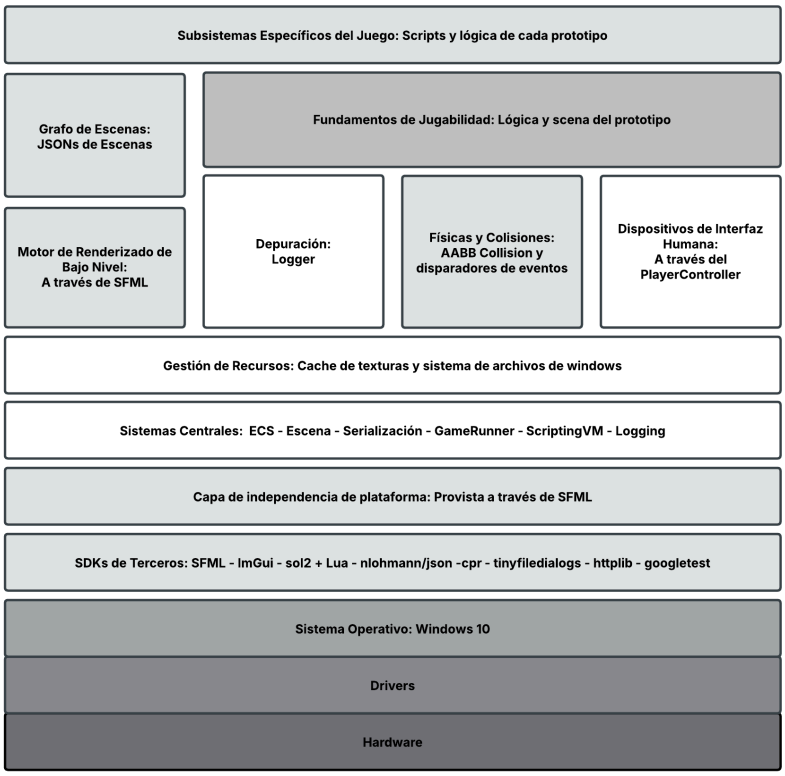


Figura 24: arquitectura por capas del motor Game Protogen.
(Gráfico de elaboración propia, 2025).

#### 3.3.3.1. Subsistemas específicos del juego

Son la “capa creativa” del motor: encapsulan reglas, eventos y comportamientos propios de cada prototipo. Se implementan a través de Lua y corren por entidad/escena, consumiendo la API pública del motor para consultar o alterar componentes y disparar el flujo del juego.

#### 3.3.3.2. Fundamentos de Jugabilidad

Orquesta el “game loop” a nivel motor: decide el orden y la frecuencia con que se ejecutan los sistemas de input, scripting, física y colisiones. Expone a Lua una API explícita la cual se compone de las siguientes funciones y variables que se pueden usar a través del scripting:

- this_id - Devuelve el **id** de la entidad que contiene el script.
- gameReset() - Comienza de vuelta la escena.

- on_spawn() - Callback que se ejecuta una sola vez al aparecer la entidad que contiene el script.
- on_update(dt: float) - Callback que corre en cada fotograma del prototipo. **dt** es el parámetro que refiere al tiempo entre el anterior fotograma y el presente.
- on_trigger_enter(other_id) - Callback que se ejecuta cuando una entidad colisiona con la que tiene el script y es disparadora de eventos. Es decir, no rige según el sistema de físicas.
- ecs.create() - Crea una nueva entidad en la escena y devuelve su **id**. Útil para agregar objetos en runtime.
- ecs.destroy(id: int) - Destruye la entidad indicada (y todos sus componentes asociados). Si el id no existe, la operación no hace nada.
- ecs.first_with(comp: string) - Devuelve el **id** de la primera entidad que tenga el componente solicitado ("Transform", "Sprite", "Collider", "Physics2D", "PlayerController"). Si no hay ninguna, devuelve 0.
- ecs.get(id: int, comp: string): Devuelve los datos del componente especificado. **comp** puede ser: “Transform”, “Sprite”, “Collider”, “Physics2D”, “PlayerController”.
- ecs.set(id: int, comp: string): Edita las propiedades del componente especificado. **comp** puede ser: “Transform”, “Sprite”, “Collider”, “Physics2D”, “PlayerController”.

#### 3.3.3.3. Grafo de Escenas

Define el modelo de entidades y componentes del mundo 2D: entidades como IDs y componentes como tablas. Funciona a través de una estructura de datos en formato JSON que puede ser cargada hacia el motor y guardada desde el motor.

#### 3.3.3.4. Motor de Renderizado de Bajo Nivel

Responsable de convertir el estado lógico en píxeles a partir de Sprite/Textura y aplica transformaciones de posición, escala y rotación. Está aislado en un sistema (Renderer2D) para poder evolucionar sin modificar otros sistemas.

### 3.3.3.5. Depuración

Provee trazas uniformes desde runtime y editor para entender qué pasa en los sistemas a través del sistema de Logs que escriben en la consola del motor la información.

### 3.3.3.6. Físicas y Colisiones

Actualiza velocidades y posiciones con gravedad y resuelve colisiones estáticas AABB con plataformas, corrigiendo penetración.

AABB ( Axis-Aligned Bounding Box ) es un algoritmo de detección de colisiones para “cajas” alineadas a ejes. Básicamente detecta una colisión si un cuadrado A se solapa con un cuadrado B en el eje X e Y. A continuación se presenta una figura que lo muestra de forma gráfica.

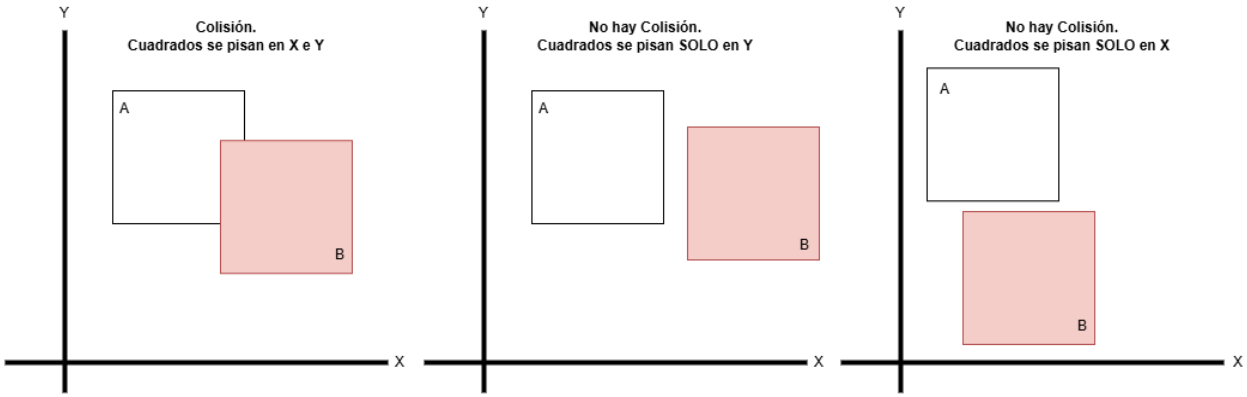


Figura 25: funcionamiento del algoritmo de detección de colisiones AABB. (Gráfico de elaboración propia, 2025).

### 3.3.3.7. Dispositivos de Interfaz Humana

Traduce entradas del teclado en intención del jugador (mover, saltar) afectando el sistema de Físicas. Esto sucede a través del PlayerController asociado a una entidad, permitiendo que el jugador la controle.

.

#### 3.3.3.8. Gestión de Recursos

Resuelve rutas (absolutas/relativas) y utiliza el sistema de archivos de windows para elegir carpeta de exportación de ejecutable jugable por ejemplo. En ejecución, la caché de texturas reduce la presión de disco y favorece tiempos de iteración.

#### 3.3.3.9. Sistemas Centrales

Los sistemas centrales representan al núcleo del motor **Entity Component System (ECS)** para almacenar estado, Serialización para persistirlo, **GameRunner** para el orden de sistemas y el ciclo de vida de juego, **ScriptingVM** como host seguro con API de métodos acotada, y **Logging** transversal. Mantener estos módulos separados facilita pruebas unitarias y evita deuda de acoplamiento.

#### 3.3.3.10. Capa de independencia de plataforma

Abstrae ventana, eventos, tiempo y gráficos básicos con la librería SFML, además de integrarse ImGui, la librería de interfaz del usuario. Permite que el motor sea agnóstico al sistema operativo a pesar de que el alcance haya sido acotado a Windows 10.

#### 3.3.3.11. SDKs de Terceros

Son los pilares de la arquitectura.

- SFML: Capa de independencia de plataforma: crea la ventana, maneja eventos de teclado/mouse, reloj y renderiza 2D.
- ImGui y ImGui-SFML (UI del editor): Construye la interfaz del editor y la integra con SFML.
- sol2 + Lua (scripting de gameplay): Embebe Lua y expone el API del motor.
- nlohmann/json (serialización): Guarda/carga escenas a JSON
- cpr / httplib / cppcodec / picosha2: Librerías HTTP para la comunicación con el servidor y la autenticación de los usuarios.
- tinyfiledialogs (diálogos nativos): Abre selectores nativos mínimos de carpetas del sistema de Windows

- googletest (tests): Marco de pruebas unitarias headless para validar la funcionalidad.

### 3.3.4. Arquitectura de despliegue cloud

Respecto a la arquitectura de despliegue en la nube, al principio se decidió diseñar una arquitectura en Amazon Web Services (AWS) debido a que es el proveedor líder (Alessandro Galimberti et al., 2025) y el autor de este documento tiene previa experiencia trabajando con él. Sin embargo, luego de un análisis, se decantó por Azure debido a su ecosistema de inteligencia artificial llamado Azure AI Foundry que simplifica la arquitectura. Entonces primero se describe la arquitectura inicial planteada y luego, la arquitectura final en Azure.


Figura 26: Se observa AWS líder dentro del cuadro de Gartner 2025 que muestra posicionamiento competitivo de las plataformas de nube más importantes según Gartner. (Alessandro Galimberti et al., 2025)

#### 3.3.4.1. Arquitectura inicial planteada (en AWS)

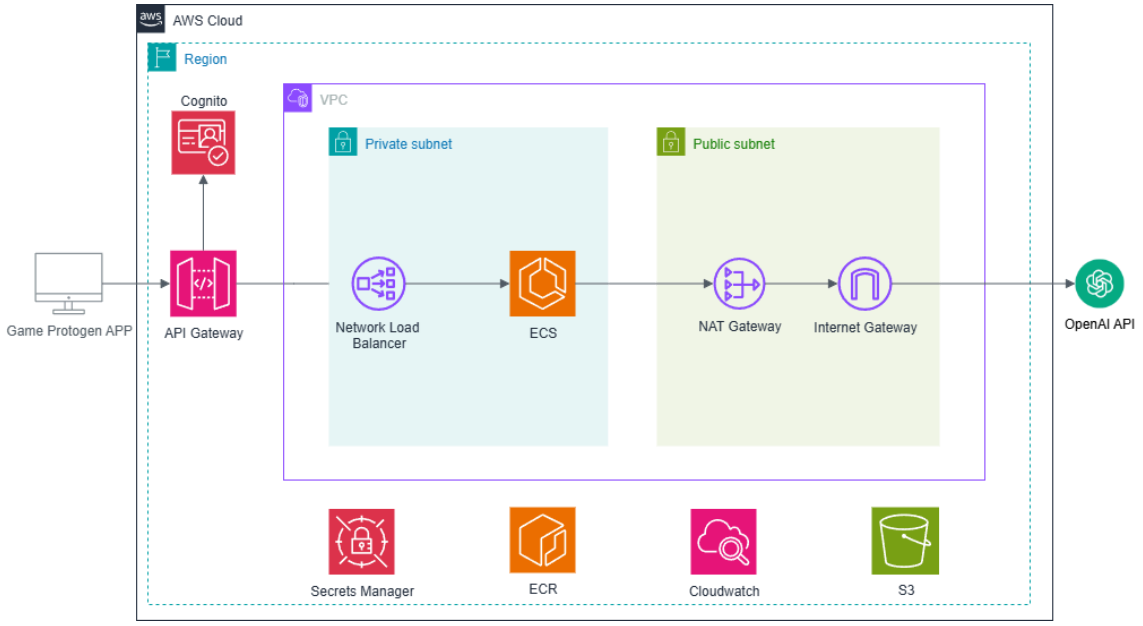


Figura 27: Arquitectura cloud AWS de Game Protogen. (Gráfico de elaboración propia, 2025).

La arquitectura inicial propuesta para Game Protogen se apoya en una VPC y segmentada en subredes públicas y privadas.

La entrada al sistema se realiza por API Gateway, que expone un endpoint con dominio propio, que valida tokens JWT emitidos por Amazon Cognito. De esta forma, sólo solicitudes autenticadas alcanzan la capa privada.

En la subred privada reside un Network Load Balancer (NLB) interno, que distribuye conexiones hacia el servicio de Amazon ECS (modo EC2) donde corre el contenedor **GameProtogenAPI**. El NLB ofrece alta performance y baja latencia, y al ser interno mantiene no expuestas las tareas del servicio. Las imágenes del contenedor se almacenan en Amazon ECR, con versionado y políticas de ciclo de vida que facilitan el despliegue continuo.

En la subred pública se ubican el NAT Gateway y el Internet Gateway, que brindan salida controlada a Internet para requisitos específicos- la API puede obtener secretos y configuraciones desde AWS Secrets Manager, emitir logs y métricas a Amazon CloudWatch, persistir snapshots/artefactos en Amazon S3 y consumir la API de OpenAI de forma segura. Este diseño mantiene los servicios de aplicación sin IP pública y restringe la superficie de exposición.

### 3.3.4.2. Arquitectura final (en Azure)

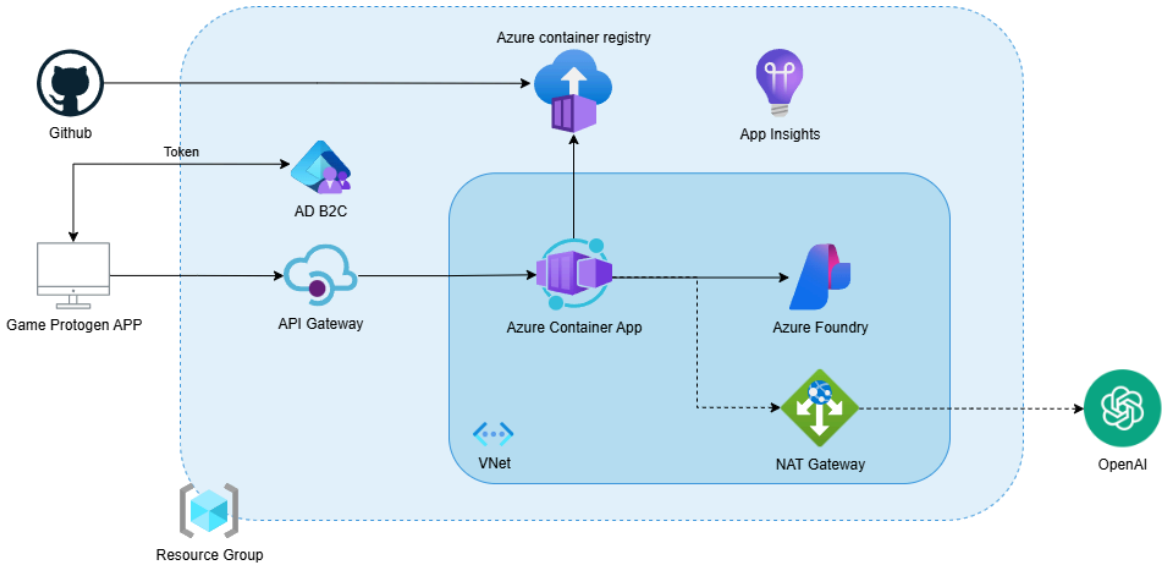


Figura 28: Arquitectura cloud AZURE de Game Protogen. (Gráfico de elaboración propia, 2025).

La arquitectura final propuesta para Game Protogen se despliega en Azure y se apoya en una VNet con subredes privadas para aislar los componentes críticos. Se eligió la región Brazil South por su cercanía con Argentina, reduciendo la latencia percibida por los usuarios y mejorando la experiencia general.

El ingreso al sistema ocurre a través de Azure API Management (API Gateway), que publica un endpoint propio y actúa como primera barrera: válida JWT emitidos por Azure AD B2C, aplica políticas (rate-limit, CORS, headers, quotas) y solo reenvía tráfico autenticado hacia la red privada. Azure AD B2C centraliza el registro/inicio de sesión de usuarios finales y simplifica el gobierno de identidades externas.

Dentro de la VNet corre Azure Container Apps (ACA), donde vive el contenedor GameProtogenAPI bajo un plan serverless que escala automáticamente por demanda. Las imágenes se almacenan en Azure Container Registry (ACR); el pipeline de GitHub Actions construye y publica a ACR y ACA hace pull seguro desde allí usando identidad administrada.

Para el consumo de IA, la API se integra con Azure AI Foundry. Cuando es posible, la comunicación se realiza mediante endpoints privados; si se necesita salida a Internet, el tráfico egrese por un NAT Gateway con IP fija para permitir rangos específicos en servicios externos sin exponer recursos internos. Sin embargo, cabe aclarar que la línea de puntos representa la llamada a OpenAI para la generación de imágenes. Esto se puede lograr a través de Azure Foundry directamente pero al momento de entrega de la tesis no se pudo conseguir cuota del mismo modelo en el servicio debido a las políticas de Azure. De esta forma, se plantea que a futuro la arquitectura pueda obviar el NAT Gateway y ahorrar costos en ese sentido.

La observabilidad se resuelve con Azure Application Insights, que recolecta métricas, logs y trazas distribuidas de la API y del runtime de contenedores para diagnóstico y tuning de performance.

En conjunto, la solución usa servicios PaaS totalmente gestionados dentro de una red privada, con control de acceso basado en identidad, alta disponibilidad, escalado automático y una postura de seguridad por diseño al minimizar la superficie pública y fijar el egreso con NAT.

### 3.3.5. Interfaz de usuario

En esta sección, se presenta la capa de presentación de la aplicación, desde su estructura mediante wireframes hasta la interfaz del producto.

#### 3.3.5.1. Wireframe

Principalmente, Game Protogen consta de 4 componentes principales:

- En la parte superior, por un lado están los operadores con los cuales se puede mover y transformar entidades. Por otro lado, se encuentran los botones para poner en marcha el prototipo o pausarlo para poder editarlo.

- Al lado derecho se posiciona un componente con dos secciones. La primera es el inspector, a través del cual el usuario puede editar las entidades, cambiando el color por ejemplo. La segunda sección es el chat con IA generativa, mediante el cual el usuario se comunica para generar cuestiones relacionadas al prototipo.
- La parte inferior consta de una consola de errores y logs que brinda información acerca del estado actual del motor con respecto a fallas de generación o en el prototipo en sí.
- En la parte central se encuentra el mundo virtual con el que el usuario interactúa utilizando los operadores. Acá es donde se renderiza el prototipo y se juega.

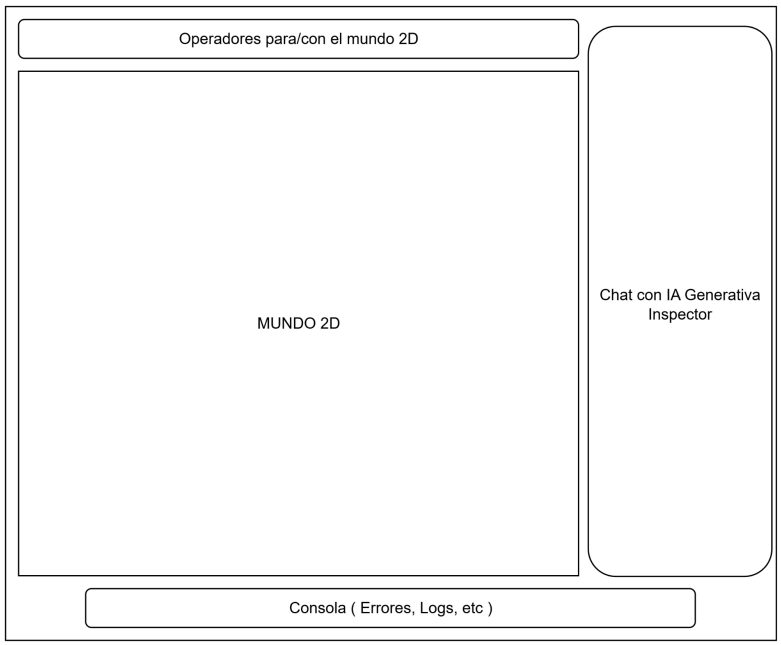


Figura 29: Wireframe estructural de Game Protogen.
(Gráfico de elaboración propia, 2025).

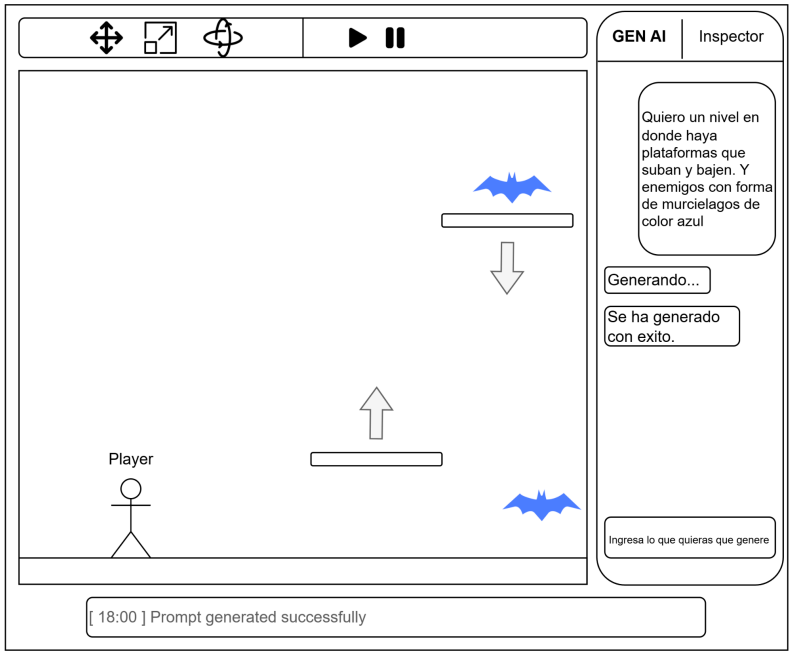


Figura 30: Wireframe de Game Protogen. (Gráfico de elaboración propia, 2025).

### 3.3.5.2. Producto

La interfaz de Game Protogen se caracteriza por tener la fuente Roboto en un tamaño acorde. Al mismo tiempo, se utilizan iconos de Angular Material, resultando en una interfaz legible y estética.

Además de los cuatro componentes esenciales de la interfaz, se observa en la parte superior un menú a través del cual el usuario puede interactuar con opciones del motor (Figura 33).

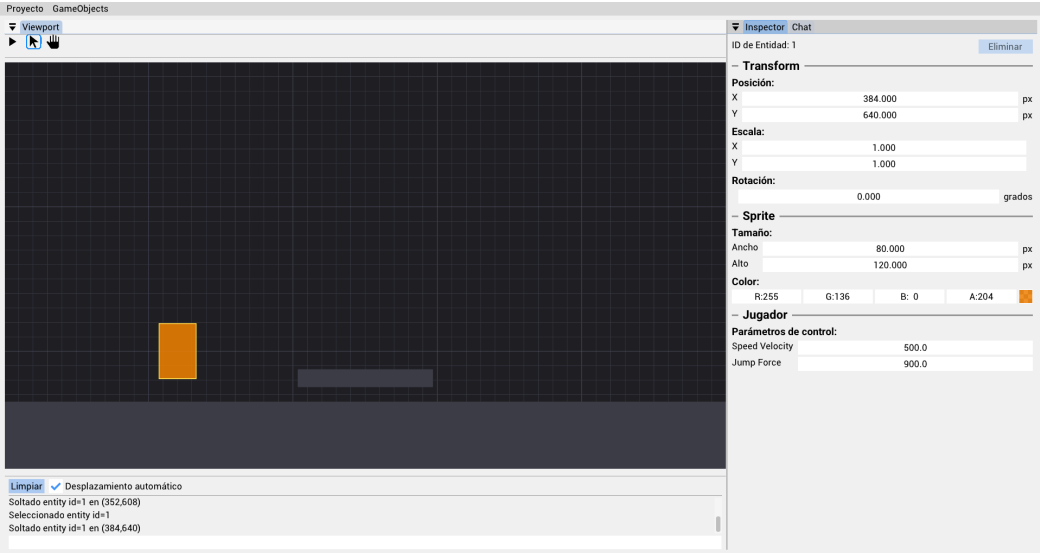


Figura 31: Interfaz de Game Protogen, con la pestaña del inspector seleccionada. (Gráfico de elaboración propia, 2025).

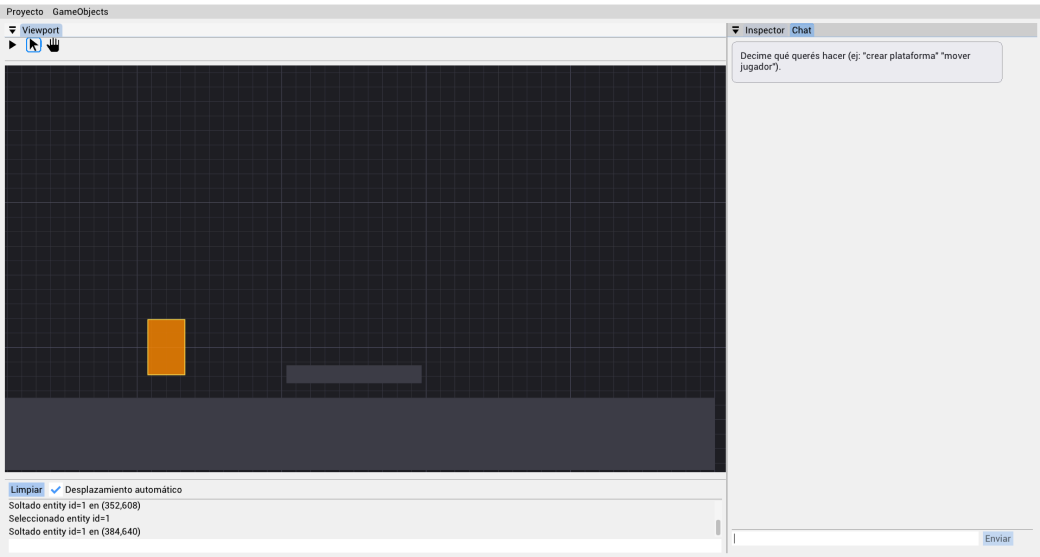


Figura 32: Interfaz de Game Protogen, con la pestaña del chat seleccionada. (Gráfico de elaboración propia, 2025).


Figura 33: Menú de opciones de Game Protogen. (Gráfico de elaboración propia, 2025).

## 3.4. Modelo de Negocio

Para **Game Protogen** se plantea un modelo de negocios Freemium con monetización a través de suscripciones. No solo esto presenta un esquema competitivo para el mercado, ya que otras soluciones como Unity Muse o Ludo.AI funcionan bajo enfoques similares, sino que además, por la naturaleza misma de utilizar servicios de inteligencia artificial que se cobran mensualmente por uso. En este sentido, la suscripción recurrente constituye el mecanismo más adecuado para garantizar sostenibilidad económica en el flujo de ingresos.

De esta manera, se plantean tres posibles planes:

- Plan Gratuito, a través del cual los usuarios podrán probar la herramienta pero con usos limitados
- Plan Individual, orientado a desarrolladores independientes
- Plan Equipo, recomendado a equipos de 2 o más desarrolladores

A continuación, se destacan los alcances y límites de cada plan en una tabla comparativa:

Tabla II: Comparación de planes de suscripción Game Protogen. (Elaboración propia, 2025).

| Plan Gratuito<br>$0 | Plan Individual<br>$20 | Plan Equipo<br>$18 |
|---|---|---|
| Exportar prototipo jugable CON marca de agua | Exportar prototipo jugable SIN marca de agua | Exportar prototipo jugable SIN marca de agua |
| Hasta 15 Imágenes 1024x1024 en calidad baja por mes | Hasta 1M tokens en imágenes 1024x1024 en calidad alta | Hasta 1M tokens en imágenes 1024x1024 en calidad alta |
| Hasta 200 mensajes por mes | Hasta 2M de tokens en mensajes por mes | Hasta 2M de tokens en mensajes por mes |
| Solo un proyecto a la vez | Proyectos ilimitados | Proyectos ilimitados y colaborativos |
| Sin soporte | Soporte personalizado | Soporte Personalizado |

Como se observa, los planes “individual” y “equipo” ofrecen los mismos beneficios, pero el último es 2 dólares más barato. Esto es porque conviene para el proyecto tener más ingreso promedio por cliente y se fomenta la adquisición del plan por estudios u equipos mediante ese descuento.

Los precios están dados particularmente por la inversión en los modelos de OpenAI por cada cliente, la arquitectura cloud y el sueldo de un desarrollador full stack.

### 3.4.1. Clientes

Game Protogen apunta a tres perfiles de clientes principales:

- Estudiantes universitarios en carreras relacionadas con la industria de los videojuegos.
- Desarrolladores indie de videojuegos.
- Equipos de empresas indie de videojuegos.

Esto es debido a que con el producto, los estudiantes pueden materializar sus ideas en prototipos jugables, sin la necesidad de tener conocimientos de programación. Por otro lado, los

profesionales pueden beneficiarse de la reducción en los tiempos de prototipado. A continuación se exponen 2 User Personas, herramientas visuales para entender de forma concisa qué tipo de usuarios utilizarían el producto.


Figura 34: User Persona de un estudiante de arte para videojuegos. (Gráfico de elaboración propia, 2025).


Figura 35: User Persona de un desarrollador de videojuegos indie. (Gráfico de elaboración propia, 2025).

### 3.4.2. Análisis FODA

El análisis FODA organiza en un marco único las fortalezas de la propuesta, las oportunidades externas, las debilidades operativas y las amenazas del entorno. Su propósito es ofrecer una lectura sintética para apoyar decisiones de alcance, crecimiento y mitigación de riesgos.

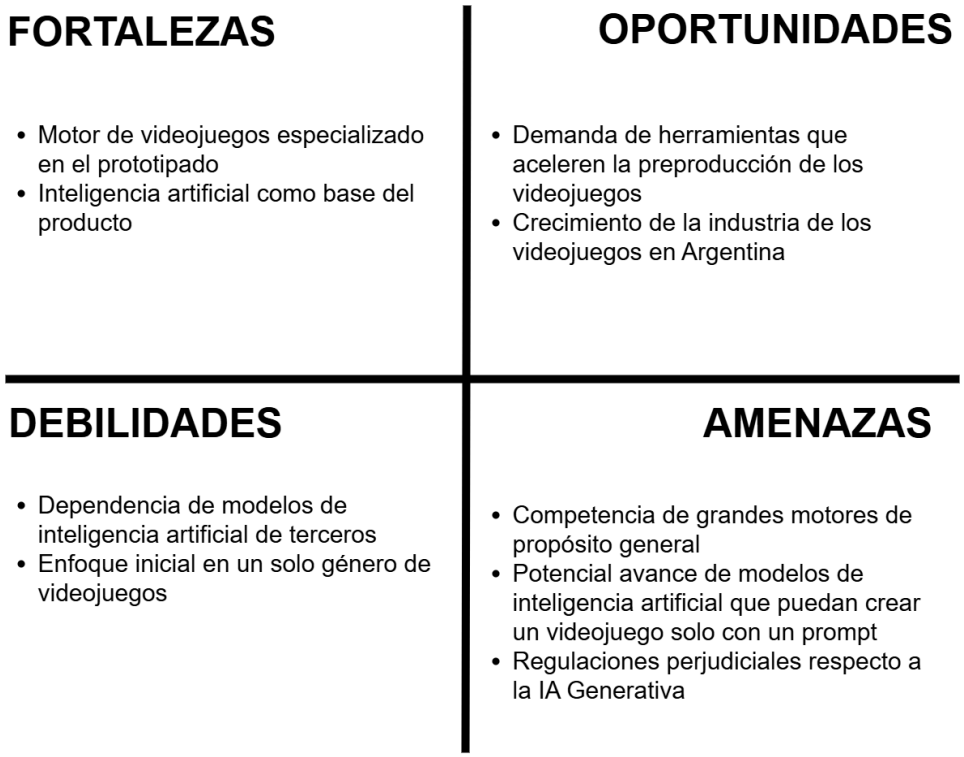


Figura 36: Análisis FODA de Game Protogen. (Gráfico de elaboración propia, 2025).

### 3.4.3. Modelo Canvas

Con el fin de resumir y finalizar este apartado, se expone a continuación un Business Model Canvas, una herramienta visual y concisa para analizar el modelo de negocios de un producto

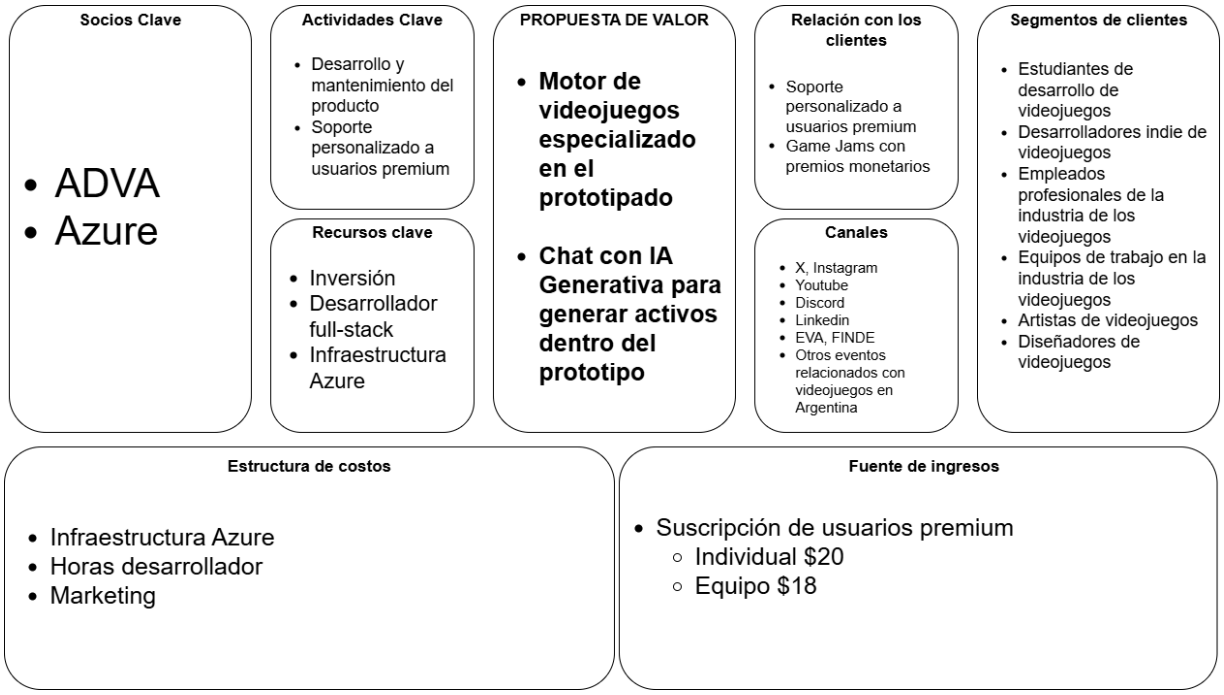


Figura 37: Modelo Canvas de Game Protogen. (Gráfico de elaboración propia, 2025).

## 3.5. Análisis Financiero

Para evaluar de forma financiera al producto, se utilizan dos métodos principales: la VAN (Valor Actual Neto) y la TIR (Tasa Interna de Retorno). Para poder realizar los cálculos, se desarrollaron flujos de fondos para tres escenarios: Pesimista, Neutro y Optimista. Estos tres varían en función de la cuota de mercado ganada en el primer año y el porcentaje de crecimiento a lo largo de los años. Se tiene en cuenta que la previsión es a 5 años, la tasa de descuento es de 9,722%, ya que está compuesta de la tasa de los bonos libres de riesgo a 5 años de Estados Unidos, siendo esta 3,722% al día 8 de octubre de 2025 («Datos históricos del bono Estados Unidos 5 años - Investing.com», 2025) y una prima de riesgo del 6% implícita de una inversión en software; y, por último, una inversión inicial de USD 1589,81 desglosada en la tabla III.

Respecto a los usuarios, se considera que actualmente existen 1686 profesionales en la industria argentina de los videojuegos (Rossi et al., 2024) y 100 mil estudiantes de carreras relacionadas con los videojuegos («El Ministerio de Desarrollo Productivo destina $250 millones para fortalecer la industria de los videojuegos», 2022). Por último, se estima que el 40% de los empleados profesionales iniciales adquieren planes individuales y el 60% planes de equipo. Esta estimación se da frente a la posibilidad de que un equipo no quiera suscribirse pero personas individuales dentro del mismo si lo hagan. Con base en estos datos, cada escenario plantea una presencia distinta en el mercado inicial y un crecimiento progresivo de usuarios, respaldado por las inversiones continuas en marketing. En el anexo C se puede observar el flujo de fondos de cada escenario particular. Se toma el valor del dólar a 1424,73 pesos argentinos.

Tabla III: Desglose de inversión inicial. (Elaboración propia, 2025).

| | |
|---|---|
| Horas desarrollador Full-stack | USD 498 |
| Computador de Trabajo | USD 600 |
| OpenAI | USD 20 |
| Arquitectura Cloud | USD 164 |
| Socio Micro ADVA | USD 301,81 |
| **Total** | **USD 1583,81** |

### 3.5.1. Escenario Pesimista

En este escenario, se consigue una cuota de mercado inicial de 25 empleados profesionales y 100 estudiantes, resultando en 110 planes individuales y 15 planes equipo. Además se estima un crecimiento del 10% anual en usuarios individuales y 20% anual en usuarios del plan equipo. La VAN da un valor negativo, presentando que el proyecto no sería rentable en este caso. La TIR por su parte no se puede calcular debido a que los flujos de fondos netos dan negativos. A continuación, se detallan estos datos de manera gráfica.

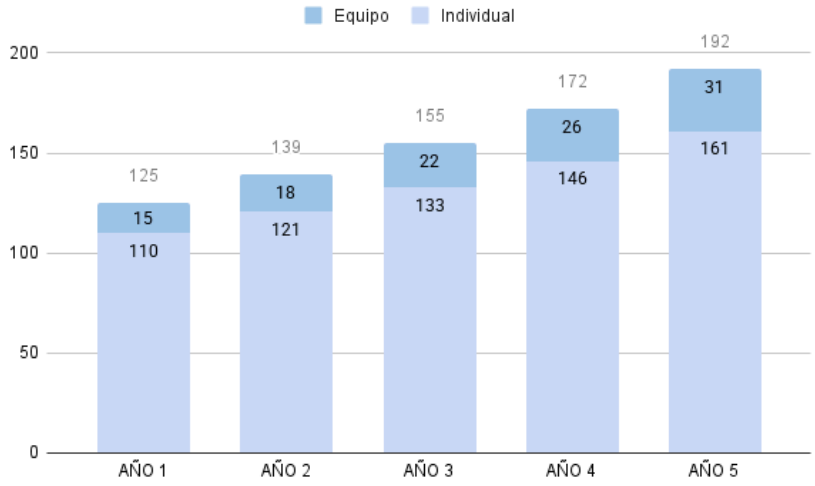


Figura 38: Crecimiento de usuarios en el escenario pesimista. (Gráfico de elaboración propia, 2025).

Tabla IV: Evaluación del escenario pesimista. (Elaboración propia, 2025).

| | |
|---|---|
| VAN | -$10.294,56 |
| TIR | Ø |

### 3.5.2. Escenario Neutro

Este caso se caracteriza por tener una participación inicial en el mercado con 34 empleados profesionales y 200 estudiantes produciendo 214 planes individuales y 20 planes equipo. El primer grupo cuenta con un crecimiento anual del 40% y el segundo, de 35%. En este escenario, la VAN tiene un valor positivo, y la TIR resulta en 30% denotando una rentabilidad positiva por el proyecto. Los siguientes gráficos ilustran estos datos.

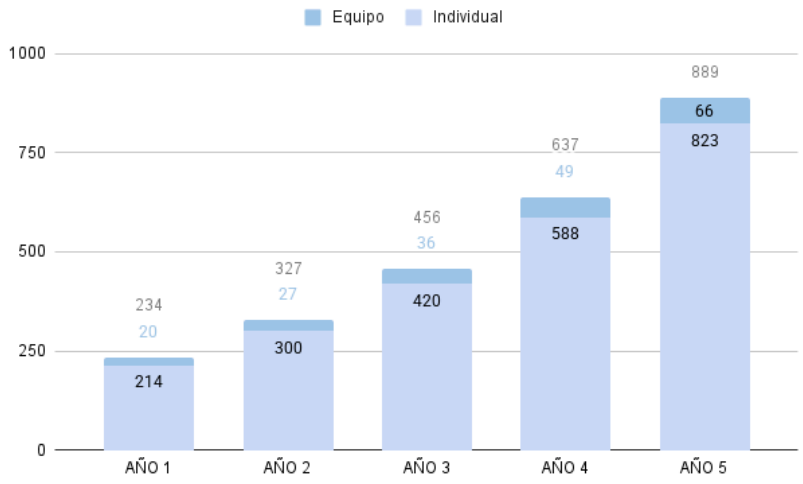


Figura 39: Crecimiento de usuarios en el escenario neutro.
(Gráfico de elaboración propia, 2025).

Tabla V: Evaluación del escenario neutro. (Elaboración propia, 2025).

| | |
|---|---|
| VAN | $2.345,51 |
| TIR | 30% |

### 3.5.3. Escenario Optimista

Por último, respecto al marco optimista, se estima obtener 42 empleados profesionales y 250 estudiantes. Esto deriva en 267 planes individuales al año 1 y 25 usuarios del plan equipo en el mismo periodo. Las ganancias que surgen a partir de estas premisas resultan en una

rentabilidad considerable por parte del proyecto. En las figuras siguientes se exponen los datos representados gráficamente.

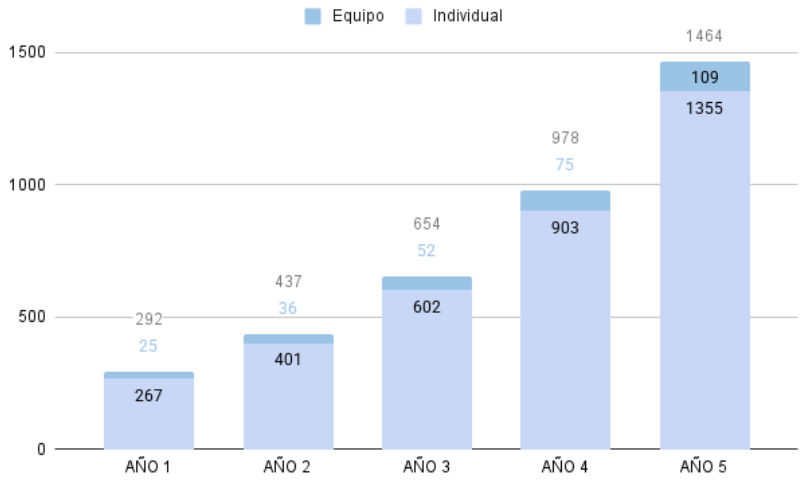


Figura 40: Crecimiento de usuarios en el escenario optimista. (Gráfico de elaboración propia, 2025).

Tabla VI: Evaluación del escenario optimista. (Elaboración propia, 2025).

| | |
|---|---|
| VAN | $11.203,53 |
| TIR | 93% |

### 3.5.4. Análisis de Resultados

En conclusión, como se observa en la tabla VII, solo en los escenarios neutro y optimista se consigue una rentabilidad sobre el proyecto. Partiendo de la base de que en los tres escenarios, la participación inicial en el mercado es ciertamente reservada, aunque justificada debido al marketing, se espera un rendimiento considerable de este proyecto.

Tabla VII: Tabla comparativa de las evaluaciones en todos los escenarios. (Elaboración propia, 2025).

| Escenario | VAN | TIR |
|---|---|---|
| Pesimista | -$8.750,94 | Ø |
| Neutro | $3.889,13 | 46% |
| Optimista | $12.747,15 | 115% |

## 3.6. Marca

La marca de la aplicación es importante dado que su función es facilitar una asociación inmediata de la herramienta con su propósito. En este sentido, A continuación se describe la justificación del logotipo y la denominación de la aplicación.

### 3.6.1. Logo

El logo de Game Protogen se caracteriza por mostrar a un cubo, que es un símbolo utilizado para representar a los motores de videojuegos en la industria, dentro de un engranaje. Este último hace referencia a la acción de prototipar. En conjunto, el logo describe a simple vista de qué se trata la aplicación.

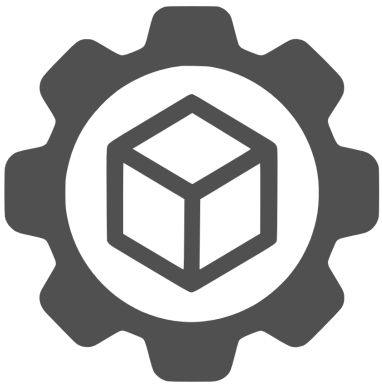

Figura 41: Icono Game Protogen. (Gráfico de elaboración propia, 2025).

### 3.6.2. Nombre

Game Protogen es el nombre elegido para esta aplicación, dos palabras que explicitan de lo que se trata la misma de forma clara y concisa. Por un lado, *Game* hace referencia al ámbito de los videojuegos. Por otro lado, *Protogen* surge como una combinación de los términos en inglés *Prototype* y *Generation*, lo cual refiere directamente al objetivo principal de la herramienta: la generación de prototipos. Esta elección nominal busca transmitir de forma clara y concisa el propósito de la aplicación, favoreciendo tanto su identificación como su posicionamiento dentro del área de desarrollo de software orientado a videojuegos.

## 3.7. Marketing

El marketing es un proceso social y directivo mediante el que los individuos y las organizaciones obtienen lo que necesitan y desean a través de la creación y el intercambio de valor con los demás (Kotler y Armstrong, 2003). En este sentido, la estrategia de marketing del presente proyecto no se limita únicamente a la promoción del producto, sino que busca generar reconocimiento y confianza en el mercado, posicionando la marca como una solución innovadora dentro de su segmento. Esto se logra a través del marketing directo e indirecto.

### 3.7.1. Marketing Directo

A través de este tipo de marketing se busca llegar directamente a los clientes (actuales o potenciales) para promocionar productos o servicios. Desde Game Protogen se plantea utilizar Google Ads para lograr este objetivo. Esta plataforma promete llegar a millones de personas con campañas de máximo rendimiento en la búsqueda, YouTube, Gmail y más («Consigue clientes y aumenta tus ventas con la publicidad en línea - Google Ads», [sin fecha]). De esta forma se posibilita alcanzar una considerable cuota del mercado.

### 3.7.2. Marketing Indirecto

Por otro lado, se plantea hacer mucho énfasis en el marketing indirecto pues se puede entender como la vibra que la empresa emite a sus clientes y los influencia tanto o más que una campaña publicitaria.

Para empezar, se propone una asociación con la Asociación De Videojuegos Argentina (ADVA), una institución intermedia que articula acciones entre los estudios y emprendedores dedicados al desarrollo de videojuego y el sector público, el sector privado y la academia, impulsando y potenciando el desarrollo de la industria local de videojuegos («Conocenos», [sin fecha]). A través de la asociación, se obtienen beneficios importantes como prensa, difusión y networking que dan una ayuda significativa en cuanto al reconocimiento local.

Figura 42: Logo de ADVA.
(«Conocenos», [sin fecha])

Luego, se sugiere organizar charlas en el marco de la Exposición de Videojuegos Argentina (EVA), un evento dedicado 100% al desarrollo de videojuegos en Latinoamérica y organizado por ADVA («EVA», [sin fecha]). Dictar conferencias en el nombre de Game Protogen puede contribuir al hecho de que la aplicación se vuelva de interés popular dentro de la industria. Además, es posible reservar un stand dentro del evento como forma de promoción y refuerzo de la presencia de la marca en el ámbito.

Figura 43: Logo de la EVA en
2025. («EVA», [sin fecha])

Por último, se plantea realizar game jams anuales. Las game jams son un tipo de hackathon en donde los participantes desarrollan un videojuego con tiempo limitado, usualmente 48 horas bajo una temática. En este caso, la game jam tiene la regla de utilizar solo Game Protogen como motor. Los mejores videojuegos creados son premiados de forma monetaria. Este tipo de actividad en función del marketing permite formar y potenciar la comunidad detrás de la aplicación.


Figura 44: Logo de la Game Protogen Jam. (Gráfico de elaboración propia, 2025).

# 4. Metodología de desarrollo

Respecto al proceso de desarrollo, se utilizó una metodología ágil para la administración del proyecto, y git para el control de versiones. A continuación se describe de qué forma se manejó cada uno y qué beneficios se obtuvieron con estos.

## 4.1. Gestión del proyecto

Para la organización y gestión del desarrollo del proyecto se utilizó la metodología Kanban, un enfoque ágil basado en la visualización del flujo de trabajo y en la entrega continua de valor.

Las principales ventajas de utilizar Kanban en este proyecto fueron:

- Flexibilidad: se adaptó fácilmente a los cambios en los requerimientos y en la priorización de tareas.
- Iteración continua: permitió trabajar en incrementos pequeños y constantes
- Transparencia: brindó una visión clara del progreso
- Foco en la productividad: al limitar la cantidad de tareas en progreso, ayudó a mantener un flujo de trabajo ordenado.

El tablero Kanban permitió organizar tanto las tareas técnicas (motor, API, integración de IA, exportación) como las tareas de documentación (encuestas, entrevistas, redacción de informe).

De esta forma, la metodología Kanban aportó una gestión visual, simple y efectiva.

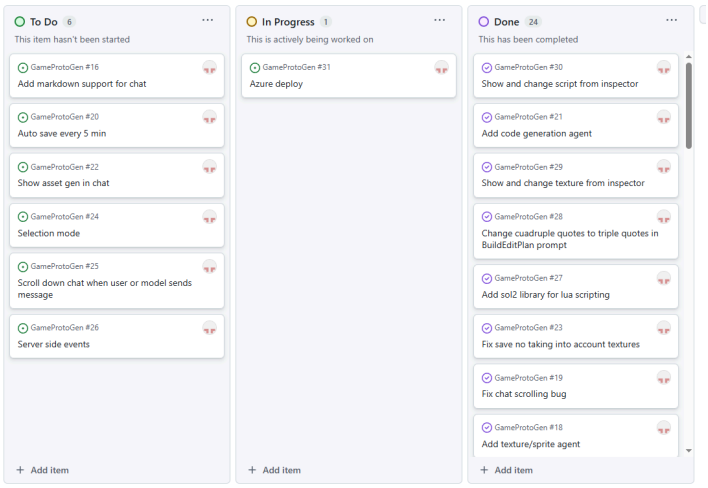


Figura 45: Tablero Kanban del desarrollo. (Imagen capturada de github projects, 2025).

## 4.2. Control de versiones

Para el manejo de las versiones e incorporación de nuevos cambios, se utilizó git y github. Se trató de usar buenas prácticas, por ejemplo la mayoría de commits siguen un formato en donde se declara con que número de tarea del tablero kanban está relacionada y, luego, el respectivo título del commit. Esto trae consigo diversos beneficios:

- Trazabilidad**:** cada cambio en el código queda vinculado a una tarea del tablero, facilitando el seguimiento desde el requerimiento hasta su implementación y despliegue.
- Auditabilidad y cumplimiento**:** el historial de commits permite reconstruir decisiones y justificar cambios.
- Colaboración estructurada**:** no solo permite mantener un proceso de desarrollo organizado, también facilita la incorporación de nuevas personas al proyecto.

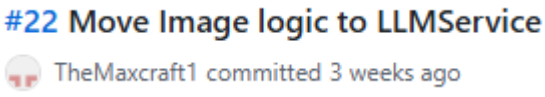


Figura 46: Ejemplo de commit. (Imagen tomada del historial de commits, 2025).

# 5. Pruebas realizadas

Con el objetivo de asegurar la calidad del producto, se realizaron distintas pruebas. En primera instancia, se hizo una prueba con un potencial cliente para validar la experiencia de usuario y afrontar mejoras en ese aspecto.

Luego, se destacan las pruebas de desarrollo de software, características de cualquier producto de software. Estas tienen como fin reducir riesgos de regresión y validar la funcionalidad de la aplicación.

Por último, se exploraron distintos modelos de inteligencia artificial para encontrar el mejor en términos de precio-calidad para este proyecto.

## 5.1. Pruebas con usuarios

El día 2 de noviembre se realizó una prueba con un usuario en vivo a través de la plataforma de comunicaciones Discord. Este usuario se caracteriza por estar dentro de uno de los segmentos que Game Protogen busca captar. Sebastian Molinari es una persona que se muestra como una persona a la que le apasionan los videojuegos y con muchos intereses creativos. Sin embargo, Sebastian no posee conocimientos de programación ni de desarrollo de videojuegos. El fin de esta prueba fue medir la experiencia de usuario y la funcionalidad de la aplicación buscando retroalimentación del usuario guiandolo por el programa.

A través de la aplicación de escritorio, el usuario interactuó con todos los flujos posibles. Particularmente, se lo noto entusiasmado con el hecho de poder generar activos usando inteligencia artificial y además, poder estructurar el nivel a su gusto. De esta forma, se puede concluir que Game Protogen le resultó satisfactoria e intuitiva en términos de experiencia de usuario. Por otro lado, Sebastian reclamó la falta de una herramienta para seleccionar varias entidades a la vez y, al mismo tiempo, una forma de identificar qué elementos específicos editar con inteligencia artificial. En consecuencia, se realizaron tareas para incorporar estos pedidos al programa.

## 5.2. Pruebas de desarrollo de software

Para validar la lógica interna de los componentes y el funcionamiento, se desarrollaron algunas pruebas unitarias y de integración entre el servidor y el motor.

### 5.2.1. Motor

A continuación, las pruebas del motor Game Protogen. En particular, la prueba **T-C001** representa una prueba unitaria, mientras que los demás, son pruebas de integración de componentes.

Tabla VIII: Pruebas de desarrollo de software del motor. (Elaboración propia, 2025).

| ID | Característica | Descripción | Resultado Esperado | Resultado de la prueba |
|---|---|---|---|---|
| T-M001 | Física de colisiones | Simula la caída de un actor con Physics2D sobre una plataforma estática y verifica *onGround=true* sin penetración. | *onGround=true* y corrección vertical tras resolver AABB. | Ok |
| T-M002 | Orquestación del ciclo jugable | Verifica que el ciclo mantenga el orden de sistemas (input→scripts→física→colisiones). | Orden estable sin efectos secundarios inesperados. | Ok |
| T-M003 | Serialización de escenas | Guarda a JSON y recarga; compara valores de componentes (Transform/Sprite/Physics). | Valores conservados | Ok |
| T-M004 | Scripting Lua | Ejecuta on_spawn y on_update en un script asociado a una entidad con binding de escena. | Ambos hooks corren sin error y pueden mutar estado. | Ok |

### 5.2.2. API

En el caso de la API, se desarrollaron dos categorías de pruebas de integración. La primera “offline” orientada a validar el contrato de las llamadas al servidor con imitaciones (mocks) de los servicios. La segunda categoría es “online” ya que se conecta con los modelos de inteligencia artificial para probar su correcto funcionamiento.

#### 5.2.2.1. Pruebas de integración offline

Tabla IX: Pruebas de integración offline de la API. (Elaboración propia, 2025).

| ID | Característica | Descripción | Resultado Esperado | Resultado de la prueba |
|---|---|---|---|---|
| T-API001 | Validación de entrada | Llamada POST /api/chat/command con parámetro prompt vacío debe fallar validación. | HTTP 400 Bad Request. | Ok |
| T-API002 | Edición de escena | Envía prompt de edición + escena mínima; debe devolver paquete de ops (incluye spawn_box). | 200 OK y JSON con "kind":"ops" y op:"spawn_box". | Ok |
| T-API003 | Q&A de diseño | Prompt de diseño (sin escena) debe retornar "kind":"text". | 200 OK con "kind":"text". | Ok |
| T-API004 | Health check | Llamada GET /health valida si el servidor está corriendo. | 200 OK y cuerpo contiene "Healthy". | Ok |
| T-API005 | Autenticación | Llamada POST /api/chat/command sin encabezado de autorización. | 401 Unauthorized. | Ok |

#### 5.2.2.2. Pruebas de integración online

Tabla X: Pruebas de integración online de la API. (Elaboración propia, 2025).

| ID | Característica | Descripción | Resultado Esperado | Resultado de la prueba |
|---|---|---|---|---|
| T-API006 | Generación de assets (sprite) | Llama a modelo de generación de imágenes y valida JSON de respuesta con sprite en base64 | JSON "kind":"asset", fileName *.png, path en Assets/Generated/, base64 decodificable. | Ok |
| T-API007 | Generación de assets (textura) | Llama a modelo de generación de imágenes y valida JSON de respuesta con textura en base64 | 200 OK y JSON con "kind":"ops" y op:"spawn_box". | Ok |
| T-API008 | Q&A de diseño | Respuesta textual no vacía a consulta de diseño | Texto no vacío. | Ok |
| T-API009 | Agentes Planner→Ops | Genera plan y lo convierte a ops JSON; valida que operaciones sea array y pase el validador | JSON con operaciones y validado | Ok |
| T-API010 | Agente Planner | Devuelve XML con <plan>...</plan>. | XML bien formado | Ok |
| T-API011 | Router multi-agente | Devuelve JSON con los agentes; si incluye asset_gen, también el modo ( sprite o textura ). | Por lo menos un agente en la lista de agentes a llamar | Ok |
| T-API012 | Generación de scripts | Genera un JSON válido con un script en lenguaje lua | Cumple todas las aserciones (regex/ausencia de require, io, os, debug, loadstring). | Ok |

## 5.3. Prueba de modelos de inteligencia artificial

Debido a la gran cantidad actual de modelos de inteligencia artificial se hicieron pruebas para concluir cuál de los modelos del mercado era más adecuado para Game Protogen. Cabe aclarar que respecto a modelos de generación de imágenes, desde un primer momento se utilizó el modelo gpt-image-1 ya que presenta una opción que los demás no, la posibilidad de generar figuras con transparencia, lo que es fundamental para generación de sprites de personajes. Por otro lado, también se debe destacar que se buscó utilizar solo modelos costo-eficientes, es decir, que posean un buen rendimiento a precio moderado. La metodología de prueba fue utilizarlos en las pruebas de integración online y además en los flujos principales de Game Protogen.

A continuación se exponen distintos modelos utilizados y las características que se notaron en función de los requerimientos de la aplicación. Se presenta a su vez para cada modelo un índice de calidad de inteligencia artificial tomado del catálogo de modelos de Azure Foundry. El mismo es un promedio (0–1) de métricas de exactitud como sobre conjuntos que evalúan razonamiento, conocimiento general, QA, matemáticas y codificación. Entonces, cuanto más cercano a 1, mejor calidad presenta el modelo. Al mismo tiempo, también se exhibe el precio estimado por cada 1 millón de tokens también relevado del mismo sitio, calculado a través de la suma del costo por tokens de entrada y el costo por tokens de salida, con una relación de 3:1 («Explore model leaderboards in Azure AI Foundry portal - Azure AI Foundry», 2025).

### 5.3.1. Modelo GPT-4o-mini

GPT-4o-mini supera a otros modelos pequeños en pruebas de referencia académicas tanto en inteligencia textual como en razonamiento multimodal. Este modelo presenta un índice de calidad de 0.72 y un coste previsto de $0.26 por 1M de tokens (Microsoft, [sin fecha]). Fue lanzado en julio de 2024 por OpenAI y se volvió popular por su bajo precio.

En las pruebas, este modelo fallaba algunas pruebas de integración y además presentaba alucinaciones, es decir, no hacía lo que se le pedía. Sin embargo, el lado más positivo de este modelo es la baja latencia de respuesta en comparación con los otros modelos probados.

### 5.3.2. Modelo Phi-4-mini-reasoning

Phi-4-mini-reasoning es un modelo abierto gestionado y ligero diseñado para el razonamiento matemático avanzado y la resolución de problemas que requieren un alto nivel de lógica. Es gestionado por Microsoft y resulta especialmente adecuado para tareas como demostraciones formales, computación simbólica y la resolución de problemas verbales de varios pasos. Gracias a su arquitectura eficiente, el modelo ofrece un equilibrio entre un alto rendimiento de razonamiento y una implementación rentable, lo que lo hace ideal para aplicaciones educativas, tutorías integradas y sistemas móviles o de borde ligero. Posee un índice de calidad de 0.69 y un precio estimado de $0.13 por 1M de tokens según el catálogo de azure foundry (Microsoft, [sin fecha]).

En las pruebas rindió mejor que GPT-4o-mini debido a que dedica más tiempo al razonamiento para luego dar una respuesta. Sin embargo, este modelo presenta una latencia similar a modelos más costo-eficientes y dió algunos fallos en las pruebas hechas.

### 5.3.3. Modelo GPT-5-mini

GPT-5-mini se describe como una versión ligera de GPT-5 para aplicaciones sensibles al costo. Como otros modelos de la familia GPT, este es ofrecido por OpenAI. Presenta como métrica de calidad 0.89, y como coste previsto $0.69 cada 1M de tokens (Microsoft, [sin fecha]).

En la aplicación, este modelo ha sido el mejor en términos de buenos resultados. No obstante, Azure Foundry no lo ofrece para la región brazil-south por lo que se está utilizando desplegado en la región east-us-2. Esto supone una latencia mayor aunque es una desventaja que vale la pena tomar debido a la eficiencia del mismo.

### 5.3.4. Modelo grok-4-fast-reasoning

Grok 4 Fast está diseñado para aplicaciones de razonamiento y de llamada de herramientas de baja latencia, destacando en IA conversacional, integraciones de API y flujos de trabajo de agentes que requieren capacidades similares a las de Grok 4 a un costo reducido. La empresa X Corp. es la que gestiona este modelo y ofrece un índice de calidad de 0.89 a la vez que un precio de $0.28 por 1M de tokens (Microsoft, [sin fecha]).

En las pruebas realizadas con este modelo, se obtuvieron buenos resultados. Azure foundry permite desplegar en brazil-south pero muchas veces el modelo no está disponible para utilizar.

### 5.3.5. Hallazgos

En conclusión, Game Protogen es una aplicación que se ve beneficiada de utilizar modelos de inteligencia artificial que utilicen el razonamiento. En particular, de los modelos probados, se destacan GPT-5-mini y grok-4-fast-reasoning. Entonces, se elige utilizar específicamente el último debido al bajo costo frente a la misma métrica de calidad que el primero y su posible despliegue en brazil-south. Sin embargo, frente a que el modelo ofrecido por X Corp a veces presenta baja disponibilidad, se actúo en consecuencia definiendo al modelo a utilizar como variable de entorno en el servidor, permitiendo utilizarlo cuando esté disponible o, en caso contrario, utilizar el modelo de OpenAI.

Por último, se procede a exhibir una tabla comparativa de los cuatro modelos probados mostrando las métricas expuestas en las anteriores secciones.

Tabla XI: Comparación entre distintos modelos probados. (Microsoft, [sin fecha]).

| | GPT-4o-mini | phi-4-mini-reasoning | Gpt-5-mini | grok-4-fast-reasoning |
|---|---|---|---|---|
| Índice de calidad | 0.72 | 0.69 | 0.89 | 0.89 |
| precio cada 1M tokens (dólares) | $0.26 | $0.13 | $0.69 | $0.28 |

# 6. Discusión

Durante el desarrollo del proyecto, han surgido distintos temas que han generado debate interno.

En primer lugar, la forma de comunicación entre la API y el motor se prestó a algunos días de pensamiento y diseño. Esto dio como resultado un protocolo de comunicación en el que se tienen en cuenta las posibles respuestas en función de los requisitos del usuario a través de su mensaje.

Una cuestión que llevó a tomar una decisión es la nube. En particular, al inicio del proyecto se tenía pensado utilizar AWS como proveedor. Sin embargo, luego de un pensamiento más profundo, se decidió elegir Azure debido a su gran apuesta por el futuro de la inteligencia artificial, lo que es la base de este proyecto. Microsoft representa todo el ecosistema de inteligencia artificial a través de la plataforma Azure Foundry, que expone los distintos modelos para poder utilizar. Uno de los problemas ligados a eso es que algunos modelos no están en la región que se utilizó en este proyecto, siendo esta *Brazil South*, por ejemplo, los últimos modelos de la familia gpt-5. Esto llevó a pensar en la utilización de otros posibles modelos, lo que provocó un trabajo de ingeniería analizando cuál era el más adecuado al producto. Por otro lado, azure foundry también pide llenar un formulario de registro para utilizar ciertos modelos, como es el caso de *gpt-image-1* que se usa en el proyecto para generar imágenes. El problema que generó esto es que el formulario pedía exclusivamente una cuenta de empresa que no se tenía en este caso. Es por esto que se decidió utilizar la api de OpenAI, no limitante, para cumplir con este requisito.

Un debate que surgió a partir de una charla con el tutor es la dependencia de plataformas que exponen modelos de inteligencia artificial. A pesar de que el alcance del proyecto y del producto no involucran el desarrollo propio de un modelo, se puede plantear la posibilidad de alojar un modelo *open-source* o incluso recurrir al *fine-tuning* para independizarse de las plataformas.

Por último, se ve un gran potencial como futura línea de desarrollo en aumentar el soporte de géneros, mejoras en sistemas de colisión, físicas, scripting y también pensar en cubrir escenarios de tres dimensiones, que por cuestiones de alcance no fueron tenidos en cuenta al igual que el sistema de sonido.

# 7. Conclusiones

Luego de todo lo analizado en este documento, validado por encuestas y entrevistas se nota un alto grado de oportunidades en cuanto a la mejora del proceso de desarrollo de videojuegos. En particular, el prototipado de videojuegos, temática que abarca este proyecto, es una etapa en donde, aun con alto impacto en tiempos y costos, no recibe la misma atención que otras etapas y rara vez se optimiza de forma deliberada. De esta forma, mediante la inteligencia artificial, que es una materia de opinión popular hoy en día, se pueden aprovechar estas oportunidades detectadas y mejorar el proceso.

En consecuencia, el presente trabajo abordó de forma exclusiva la etapa de prototipado de videojuegos haciendo uso de inteligencia artificial para optimizar el proceso. A través de un chat, que es estándar en productos de hoy en día para aplicar la IA, se observa una posible reducción de la brecha técnica. Es decir, personas ajenas a la industria de los videojuegos pueden interactuar y generar algo que sea de su gusto. Esto demuestra que la inteligencia artificial es una gran herramienta que tiene mucho potencial para democratizar conocimiento e industrias, no solo la de los videojuegos.

Entonces, en días actuales en donde los usuarios pecan de impaciencia constante debido a la inmediatez que se vive, un producto como Game Protogen que pueda facilitar y optimizar una etapa del proceso de desarrollo de videojuegos, permite no solo incrementar la velocidad de producción de un videojuego, sino que también mejorar la calidad del producto al haber pasado por un motor de videojuegos enfocado en el prototipado que ayude a descubrir el potencial de lo que puede llegar a ser un videojuego completo.

# 8. Bibliografía

ADELAIDA TRUJILLO VÁZQUEZ, CONCHI PARRA MEROÑO, y ANGEL PABLO CANO GOMEZ, 2015. Taxonomía del videojuego: un planteamiento por géneros. *Sociedad Latina de Comunicación Social* [en línea], [consulta: 7 junio 2025]. Disponible en: https://www.researchgate.net/publication/308983850_Taxonomia_del_videojuego_un_planteamiento_por_generos.

AI Assistant Hub - Godot Asset Library. [en línea], [sin fecha]. [consulta: 8 junio 2025]. Disponible en: https://godotengine.org/asset-library/asset/3427.

ALEEM, S., CAPRETZ, L.F. y AHMED, F., 2016. Game Development Software Engineering Process Life Cycle: A Systematic Review. En: arXiv:1711.08527 [cs], *Journal of Software Engineering Research and Development*, vol. 4, no. 1, pp. 6. ISSN 2195-1721. DOI 10.1186/s40411-016-0032-7.

ALESSANDRO GALIMBERTI, DENNIS SMITH, CAROLIN ZHOU, ED ANDERSON, TONY HARVEY, y DOUGLAS TOOMBS, 2025. Gartner. *Magic Quadrant for Strategic Cloud Platform Services* [en línea]. [consulta: 7 octubre 2025]. Disponible en: https://www.gartner.com/doc/reprints?id=1-2LM5HTYC&ct=250805&st=sb&trk=59e09e9f-ef2e-4f08-9d0c-fce0d64139c7&sc_channel=el.

ALHARTHI, S.A., 2025. Generative AI in Game Design: Enhancing Creativity or Constraining Innovation? *Journal of Intelligence*, vol. 13, no. 6, pp. 60. ISSN 2079-3200. DOI 10.3390/jintelligence13060060.

ANDERSEN, G., 2024. The Importance of Prototyping in Video Game Design: Testing ideas before development. [en línea]. [consulta: 1 junio 2025]. Disponible en: https://moldstud.com/articles/p-the-importance-of-prototyping-in-video-game-design-testing-ideas-before-development.

ANDREW BEGEMANN y JAMES HUTSON, 2024. Empirical insights into AI-assisted game development: A case study on the integration of generative AI tools in creative pipelines. *Metaverse*, ISSN 2810-9791. DOI 10.54517/m.v5i2.2568.

BELLMAN, R., 1978. *An Introduction to Artificial Intelligence: Can Computers Think?* S.l.: Boyd & Fraser Publishing Company. ISBN 978-0-87835-066-7.

BERZAL, F., 2018. *Redes Neuronales y Deep Learning*. S.l.: Fernando Berzal. ISBN 978-1-7312-6538-8.

BISHOP, C.M., 2006. *Pattern Recognition and Machine Learning*. S.l.: Springer. ISBN 978-0-387-31073-2.

BRAGA-NETO, U., 2020. *Fundamentals of Pattern Recognition and Machine Learning*. S.l.: Springer International Publishing AG. ISBN 978-3-030-27655-3.

Cinema - Worldwide. [en línea], 2025. S.l.: Statista. [consulta: 1 junio 2025]. Disponible en: https://www.statista.com/outlook/amo/media/cinema/worldwide.

Conocenos. *ADVA* [en línea], [sin fecha]. [consulta: 9 octubre 2025]. Disponible en: https://adva.vg/conocenos/.

Consigue clientes y aumenta tus ventas con la publicidad en línea - Google Ads. *Google Business* [en línea], [sin fecha]. [consulta: 9 octubre 2025]. Disponible en: https://business.google.com/es-all/google-ads/.

Create hit games with the power of AI with Ludo. *Ludo.ai* [en línea], [sin fecha]. [consulta: 8 junio 2025]. Disponible en: https://ludo.ai/.

Datos históricos del bono Estados Unidos 5 años - Investing.com. *Investing.com Español* [en línea], 2025. [consulta: 9 octubre 2025]. Disponible en: https://es.investing.com/rates-bonds/u.s.-5-year-bond-yield-historical-data.

DU, Y., LIU, Z., LI, J. y ZHAO, W.X., 2022. *A Survey of Vision-Language Pre-Trained Models* [en línea]. 16 julio 2022. S.l.: arXiv. [consulta: 7 junio 2025]. arXiv:2202.10936. Disponible en: http://arxiv.org/abs/2202.10936.

EGENFELDT-NIELSEN, S., SMITH, J.H. y TOSCA, S.P., 2024. *Understanding Video Games: The Essential Introduction*. S.l.: Taylor & Francis. ISBN 978-1-040-00248-3.

El Ministerio de Desarrollo Productivo destina $250 millones para fortalecer la industria de los videojuegos. *Argentina.gob.ar* [en línea], 2022. [consulta: 9 octubre 2025]. Disponible en: https://www.argentina.gob.ar/noticias/el-ministerio-de-desarrollo-productivo-destina-250-millones-para-fortalecer-la-industria-de.

EVA. *EVA* [en línea], [sin fecha]. [consulta: 9 octubre 2025]. Disponible en: https://expoeva.com/.

Explore model leaderboards in Azure AI Foundry portal - Azure AI Foundry. [en línea], 2025. [consulta: 29 octubre 2025]. Disponible en: https://learn.microsoft.com/en-us/azure/ai-foundry/concepts/model-benchmarks.

*Godot AI Assistant Hub (1/6) - What is it?* [en línea], 2024. [consulta: 8 junio 2025]. Disponible en: https://www.youtube.com/watch?v=3PDKJYp-upU.

GOODFELLOW, I., BENGIO, Y. y COURVILLE, A., 2016. *Deep Learning*. S.l.: MIT Press. ISBN 978-0-262-03561-3.

GREGORY, J., 2018. *Game Engine Architecture, Third Edition*. S.l.: CRC Press. ISBN 978-1-351-97427-1.

GUI, J., CHEN, T., ZHANG, J., CAO, Q., SUN, Z., LUO, H. y TAO, D., 2024. A Survey on Self-supervised Learning: Algorithms, Applications, and Future Trends. En: arXiv:2301.05712 [cs] [en línea], [consulta: 1 junio 2025]. DOI 10.48550/arXiv.2301.05712. Disponible en: http://arxiv.org/abs/2301.05712.

GUSTAFSSON, A., 2014. *An Analysis of Platform Game Design : Implementation Categories and Complexity Measurements* [en línea]. S.l.: Linnaeus University. [consulta: 7 junio 2025]. Disponible en: https://urn.kb.se/resolve?urn=urn:nbn:se:lnu:diva-35517.

How does everyone actually feel about the points system for Muse? - Muse. *Unity Discussions* [en línea], 2025. [consulta: 8 junio 2025]. Disponible en: https://discussions.unity.com/t/how-does-everyone-actually-feel-about-the-points-system-for-muse/333234.

Introduction. *Godot Engine documentation* [en línea], [sin fecha]. [consulta: 8 junio 2025]. Disponible en: https://docs.godotengine.org/en/stable/about/introduction.html.

JOUNI SMED y HARRI HAKONEN, 2003. owards a Definition of a Computer Game. *TUCS Technical Reports* [en línea], no. 540, [consulta: 7 junio 2025]. ISSN 1239-1891. Disponible en: https://www.researchgate.net/publication/2927941_Towards_a_Definition_of_a_Computer_Game.

KOTLER, P. y ARMSTRONG, G., 2003. *Fundamentos de marketing*. S.l.: Pearson Educación. ISBN 978-970-26-0400-6.

KURZWEIL, R., 1990. *The Age of Intelligent Machines*. S.l.: Kurzweil Foundation. ISBN 978-0-262-11121-8.

LINÅKER, J., BJARNASON, E. y FAGERHOLM, F., 2024. *Pre-Release Experimentation in Indie Game Development: An Interview Survey* [en línea]. 26 noviembre 2024. S.l.: arXiv. [consulta: 15 junio 2025]. arXiv:2411.17183. Disponible en: http://arxiv.org/abs/2411.17183.

MANKER, J. y ARVOLA, M., 2011. Prototyping in Game Design: Externalization and Internalization of Game Ideas. *Proceedings of HCI 2011 The 25th BCS Conference on Human Computer Interaction* [en línea]. S.l.: BCS Learning & Development, [consulta: 8 junio 2025]. DOI 10.14236/ewic/HCI2011.57. Disponible en: https://www.scienceopen.com/hosted-document?doi=10.14236/ewic/HCI2011.57.

MARGARIS, J., 2025. Prototyping is Often a Waste of Time. *On Video Games* [en línea]. [consulta: 8 junio 2025]. Disponible en: https://jmargaris.substack.com/p/prototyping-is-often-a-waste-of-time.

MICROSOFT, [sin fecha]. AI Model Catalog | Azure AI Foundry Models. *Azure AI Foundry* [en línea]. [consulta: 29 octubre 2025]. Disponible en: https://ai.azure.com/catalog.

MINAEE, S., MIKOLOV, T., NIKZAD, N., CHENAGHLU, M., SOCHER, R., AMATRIAIN, X. y GAO, J., 2025. *Large Language Models: A Survey* [en línea]. 23 marzo 2025. S.l.: arXiv. [consulta: 7 junio 2025]. arXiv:2402.06196. Disponible en: http://arxiv.org/abs/2402.06196.

MURPHY, K.P., 2012. *Machine Learning: A Probabilistic Perspective*. S.l.: MIT Press. ISBN 978-0-262-01802-9.

PARENTE, D., 2023. Prototipado de videojuegos: la guía definitiva para desarrolladores de juegos principiantes. [en línea]. [consulta: 1 junio 2025]. Disponible en: https://www.danielparente.net/es/2023/11/05/prototipado-de-videojuegos/.

POLITOWSKI, C., FONTOURA, L., PETRILLO, F. y GUÉHÉNEUC, Y.-G., 2016. Are the Old Days Gone? A Survey on Actual Software Engineering Processes in Video Game Industry. En: arXiv:2009.02448 [cs], *Proceedings of the 5th International Workshop on Games and Software Engineering* [en línea]. S.l.: s.n., pp. 22-28. [consulta: 8 junio 2025]. DOI 10.1145/2896958.2896960. Disponible en: http://arxiv.org/abs/2009.02448.

¿Qué es la inteligencia artificial o IA? *Google Cloud* [en línea], [sin fecha]. [consulta: 30 mayo 2025]. Disponible en: https://cloud.google.com/learn/what-is-artificial-intelligence.

RADFORD, A., NARASIMHAN, K., SALIMANS, T. y SUTSKEVER, I., 2018. Improving Language Understanding by Generative Pre-Training. ,

Real-Time 3D Development Platform & Editor. *Unity* [en línea], [sin fecha]. [consulta: 8 junio 2025]. Disponible en: https://unity.com/products/unity-engine.

ROSSI, A., FOLE, F., MARTÍN, MIGUEL, GUADALUPE DE LA IGLESIA, y GIMENA BRUNO, 2024. INFORME 2024. . S.l.: Observatorio de la industria de los videojuegos de Argentina.

RUSSELL, S.J. y NORVIG, P., 2004. *Inteligencia artificial: un enfoque moderno*. S.l.: Pearson Educación. ISBN 978-84-205-4003-0.

SAMUEL, A.L., 1959. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210-229. ISSN 0018-8646. DOI 10.1147/rd.33.0210.

SCHELL, J., 2019. *The Art of Game Design: A Book of Lenses, Third Edition*. S.l.: CRC Press. ISBN 978-1-351-80363-2.

SUDHAKARAN, S., GONZÁLEZ-DUQUE, M., GLANOIS, C., FREIBERGER, M., NAJARRO, E. y RISI, S., 2023. *MarioGPT: Open-Ended Text2Level Generation through Large Language Models* [en línea]. 8 noviembre 2023. S.l.: arXiv. [consulta: 8 junio 2025]. arXiv:2302.05981. Disponible en: http://arxiv.org/abs/2302.05981.

The PC & Console Gaming Report 2025. [en línea], 2025. S.l.: NewZoo. [consulta: 1 junio 2025]. Disponible en: https://newzoo.com/resources/trend-reports/the-pc-console-gaming-report-2025.

Unity muse price disappointment - Muse. *Unity Discussions* [en línea], 2023. [consulta: 8 junio 2025]. Disponible en: https://discussions.unity.com/t/unity-muse-price-disappointment/313263/6.

Unity Muse: Unlock Your Creative Potential with AI. *Unity* [en línea], [sin fecha]. [consulta: 8 junio 2025]. Disponible en: https://unity.com/products/muse.

VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A.N., KAISER, L. y POLOSUKHIN, I., 2023. *Attention Is All You Need* [en línea]. 2 agosto 2023. S.l.: arXiv. [consulta: 6 junio 2025]. arXiv:1706.03762. Disponible en: http://arxiv.org/abs/1706.03762.

YANG, L., ZHANG, Z., SONG, Y., HONG, S., XU, R., ZHAO, Y., ZHANG, W., CUI, B. y YANG, M.-H., 2024. *Diffusion Models: A Comprehensive Survey of Methods and Applications* [en línea]. 2 diciembre 2024. S.l.: arXiv. [consulta: 7 junio 2025]. arXiv:2209.00796. Disponible en: http://arxiv.org/abs/2209.00796.

YANG, R., DAI, J., VASILAKIS, N. y RINARD, M., 2025. *Evaluating the Generalization Capabilities of Large Language Models on Code Reasoning* [en línea]. 7 abril 2025. S.l.: arXiv. [consulta: 7 junio 2025]. arXiv:2504.05518. Disponible en: http://arxiv.org/abs/2504.05518.

# ANEXO A: TRANSCRIPCIÓN DE ENTREVISTAS

A continuación, se detallan las transcripciones limpias (omitiendo muletillas, repeticiones, trabas y preguntas no relevantes) de las dos entrevistas realizadas.

## Entrevista con Juan Pablo Cardoso (Desarrollador de Videojuego indie)

Se ha grabado la entrevista efectuada y subido al canal de youtube del autor de este proyecto con el debido permiso de Juan Pablo. Se puede visualizar a través del siguiente link: https://www.youtube.com/watch?v=Ibk3N2CyBU4

**¿Querés presentarte Juan Pablo?¿Cómo estás?**

Bueno, ¿Cómo andas? Soy Juan Pablo Cardoso, desarrollador de videojuegos y soy recibido de ingeniería en informática de la Universidad Nacional de Mar del Plata. Me estoy metiendo un poco más ahora en otras áreas, más que nada game design que me interesa bastante porque estoy haciendo mis juegos de manera independiente y creo que es una pata bastante clave para llamar la atención y que el juego llegue a la gente.

**¿Sabes cuántos proyectos de videojuegos ( de cualquier largo ) hiciste a lo largo de tu carrera aproximadamente?**

Más de 10.

**¿Qué motores usas para hacer los juegos?**

Primero arranqué aprendiendo con Godot. Después me separe un tiempo por el tema de la salida laboral y me interiorice más en Unity. Incluso saque un juego independiente en Unity. Ahora estoy volviendo a las raíces, con Godot, de nuevo, para la parte 3D que nunca había usado. El proyecto actual que estoy haciendo es de terror y le estoy metiendo pila por ahí.

**¿Cómo encaras normalmente la etapa de prototipado en un videojuego?**

Ahora con la mentalidad que tengo, trato de balancear entre el tipo de juego que me gusta, el tipo de juego que se que puedo hacer con mis habilidades técnicas y el tipo de juego que puede llegar a tener salida laboral en el mercado. Trato de hacer una conjunción de esas tres cosas. Aunque

ahora estoy más por el lado del terror, de lo que soy muy apasionado. Trato de buscar inspiración para el tema de las ideas con algún tema que me interese y a partir de ahí desarrollo una técnica que tenga que ver con ese tema. También alguna mini-historia para fundamentar un poco, no tengo mucha base en la parte narrativa. Me quiero desarrollar un poco más ahí porque es bastante clave en los juegos de terror también. Después, de ahí empezar a anotar la lista de mecánicas principales y arrancar de lo principal hasta lo más secundario para tener una versión base para ver si la jugabilidad básica llama la atención o no, es divertida o no, engancha o no, y si cumple con lo que yo pensaba porque hay muchas veces que entre la idea y lo que realmente queda en el juego es muy diferente. Y bueno, intentar encontrar lo divertido ahí. Si no sale, ir matando o virando para otro lado porque si no es divertido ya desde el inicio, por más que le agregues mucha decoración, sonido, imágenes, ya es muy complicado que se vuelva divertido de repente.

**¿Usas alguna herramienta para prototipar?¿O simplemente vas al motor de videojuegos directo?**

Soy más de mandarme al motor de videojuegos. He usado otras herramientas que son más de prototipado pero capaz más para juegos de Game Jams, como Puzzle Script, que es para prototipar juegos de puzzles sokoban que son muy específicos, entonces está optimizado para hacer ese tipo de juegos. Ahora, esa misma herramienta, como te exporta los niveles en .txt también la estoy usando para el juego que estoy haciendo ahora, para una parte que tengo un nivel de un laberinto la hice toda ahí y fui exportando el .txt al juego. Esta muy bueno porque tiene un editor de niveles que es por celda, podes ponerle distintos dibujos a cada celda, es mucho más gráfico y esta piola para prototipar ahí diseño de nivel, así, por cuadrícula.

**¿Creés que hay algo cuando prototipas que te lleva más tiempo o esfuerzo?**

Creo que lo principal, las mecánicas, que si queres que quede bien representativo de lo que buscas, lleva mucho ajuste. Prueba y error, ajustar valores. Ir probando. Justo en Godot, tienen unas herramientas buenas para eso, Unity igual también. Podes cambiar unas variables mientras ejecutas el juego, y eso te acelera mucho porque si tuvieras que frenar el juego y arrancar cada vez que querías probar algo nuevo, tardarías mucho más tiempo.

**¿Creés que hay algún desafío o frustración que tenes cuando realizas un prototipo?**

El desafío más grande creo que está, para mí, más que del lado de hacer, que siento que tengo capacidad técnica para hacer, el lado de probar. Muchas veces lo pruebo yo y, como que te encerras en tu propia burbuja y es difícil mantener la visión de los jugadores. Te comentaba hace un rato lo del juego de terror, que hay muchas cosas por sentado. Aparte cuanto más largo es el juego, bueno en el prototipo capaz no te llenas tanto con la idea vos, pero bueno, tenes algunas preconcepciones también. Está bueno tener la visión de alguien que lo agarra de cero y no entiende nada o se choca contra cosas que vos no te chocas. Está muy bueno ese ida y vuelta, más que nada en un prototipo cuando no tenes nada mostrable. Imaginate si ya cuesta conseguir jugadores para juegos que ya tienen una demo, bastante más pulido. Imagínate un prototipo que se ve horrible, no te lo va a querer jugar nadie. Un familiar o amigo capaz. Ese es el desafío, creo uno de los más grandes.

**¿Te pasó alguna vez algo como planificar un prototipo para dos semanas y que te lleve más tiempo?**

Bueno, mismo con el prototipo del juego que estoy haciendo ahora. Al principio quería arrancarlo como un juego de Game Jam y luego expandir esa idea. El alcance, muy difícil planificarlo y godot tampoco lo había usado. Pero bueno, se me fue un poco de las manos. La versión de game jam era vos caminando y agarrando las notas sin decoración ni nada. Solo hice que puedas agarrar las notas y que puedas terminar el juego. Pero sí, como que se me fue de las manos el alcance. Hay que saber que tipo de proyectos entran en una game jam y qué tipo de proyectos son para un poco de más tiempo también.

**En este sentido, ¿Hay algunas tareas específicas que decis, esto es lo que más tiempo me lleva?**

En mi caso, lo que más me llevó ahora en el juego este es la parte del diseño de niveles y también ajustar el game design, ya que nunca lo había hecho antes. Además conseguir los modelos 3D para armar el layout del nivel y que queda más o menos presentable. Y bueno después fui reemplazando algunas cosas también. Pero como que ese ida y vuelta te va sacando del flow del desarrollo y te va ralentizando.

**¿Cómo te sentís con el uso de la inteligencia artificial a la hora de desarrollar? En términos morales y de practicidad**

Mira, yo lo he probado para el arte y para temas de código. Para el código siento que esta bueno. Antes de la inteligencia artificial se buscaba en Stack Overflow y tenias que ajustarlo un poco más. Ahora está más personalizado, mismo la inteligencia artificial saca muchas cosas de esas fuentes también. La verdad que me parece muy práctico, pero tiene sus limitaciones. Obviamente no le podes decir, haceme un juego de carreras multijugador y te lo hace de una. Es como para cosas muy puntuales que vos ya tenes mas o menos una idea de lo que queres hacer o una función específica de algo. Si queres que te la revise o te la ajuste. Para eso en mi experiencia funciona muy bien. Y mismo, pasarle cosas que te las traduzca de un lenguaje a otro está muy bueno. Un caso de uso que encontré hace poco es el tema de los Shaders, que es de programación visual. Diferentes tecnologías usan diferentes lenguajes, y hay una página llamada “shader toy” que tiene muchos shaders pero no tiene el mismo lenguaje de shader que godot. Pero se lo pedís a la IA y te lo transforma bastante 1 a 1 y sale andando.

Después, el lado del arte también lo he usado pero ahí siento que hay más complicaciones, tanto morales porque bueno, es como que el artista tiene un grado más de autoría sobre el arte que el programador que capaz vos lo programas y decís, este es un texto que mande. El artista deja una parte de sí en la obra y es mucho más personal. Aparte de ese tema moral, está el tema de que las inteligencias artificiales se entrenan con arte de artistas que supuestamente no dan consentimiento o a veces les cambian los términos de uso mientras usan las cosas. Yo que no soy artista, por más que *prompteo* cosas, es muy dificil armar algo que sea coherente y que quede lindo esteticamente, que tenga bien los tonos de color que tiene que tener, que te forme una paleta de colores copada, y que tenga sentido con tu juego. Aun así usandolo, creo que es más el uso que se le puede dar desde el desconocimiento para alguna referencia para comunicar con arte en vez de con palabras que es mucho mejor. Mismo me ha pasado de contratar gente, músicos o artistas, que su idioma es el idioma de lo que hacen. Es mucho más fácil mostrar algún ejemplo o algo de ese mismo arte que intentar explicar con palabras. Y más no sabiendo los términos técnicos de lo que se habla en arte o en audio. Pero sí, hay desafíos y creo que hay oportunidades también. Hay que saber navegar en el tema IA pero tampoco negarlo al 100%. Usarlo para lo que vaya sirviendo y se vea que da buen resultado.

**¿Si existiese una herramienta que te permita crear prototipos con ayuda de inteligencia artificial generativa, es decir, te puede ayudar a generar código, sprites, texturas en la misma herramienta, la usarías?¿Funcionaría para tu proceso de desarrollo de prototipos?**

Yo pienso que puede llegar a servir. El tema es que hay desafíos ahí de hacerlo de que quede bien eso pero podría estar bueno. Por lo menos para un prototipo. Tendría la herramienta capaz que darte la habilidad de después cambiar esos assets y cosas que están hechas así nomás con IA. Reemplazarlas con algo más final o intermedio.

**En este sentido, ¿Qué funcionalidades te gustaría que tenga una aplicación de este estilo?**

Estaria bueno, como te digo, que tenga cosas parecidas a los "resources" de godot o los "scriptable objects" de unity que tienen como para parametrizar cosas, y en eso parametrizar cosas que tenga para cambiar el arte de las cosas. El arte de la IA te puede largar cosas de diferentes tamaños, tendría que estar muy estandarizado a exactamente lo que precisas. También cosas descargadas de internet, como placeholders, muchas veces pasa que te quedan de distintos tamaños y tenes que re escalarlos dentro del videojuego. Tema audio, la verdad que no he usado (inteligencia artificial) pero imagino que también lo mismo, que te de herramientas para cambiarlo fácil. El audio en videojuegos hay algunos errores que pueden pasar como que no "loopee" bien el audio cuando lo pones de fondo. El tema de los volúmenes, que hay cosas que suenan muy fuerte, otros muy bajo. Están esos desafíos que llevan trabajo manual ajustar esas cosas. Si se tiene una herramienta que genera las cosas de cero, que genere las cosas ya más consistente en ese aspecto.

**¿Pagarías por usarla?¿Preferirías una subscripción mensual, que sea gratis, licencia directamente o se te ocurre algún otro modelo de negocios que te parezca?**

Me parece que hoy en día, teniendo ChatGPT gratis, tendría que tener algo parecido. Una versión gratis con cierta cantidad de usos y, si pagas tenes la versión "full" con todas las funcionalidades.

## Entrevista con Cristian Basoalto (Director de Bacord Games)

En este caso, también se ha grabado la entrevista efectuada y se ha subido al canal de youtube del autor con el permiso de Cristian. Se puede visualizar a través del siguiente link: https://www.youtube.com/watch?v=mRiwkL4lINQ

**Cris, ¿Querés presentarte?**

Si, soy director de Bacord Games, un estudio que se fundó hace 8 años casi ya. En el medio de esos 8 años trabajamos para muchas empresas en el ambiente del gambling, para casinos y páginas de gambling. Los últimos proyectos ya nos pasamos a hacer juegos para Nintendo Switch, ya sacamos 3 juegos. También hemos hecho de intermediario entre desarrolladores y un publisher, para conseguirle publisher y terminar de pulir los juegos de ellos. Las tareas mias aca adentro son de dirección más que nada en este momento pero en realidad soy músico. Empecé haciendo sound design y después pase a ser game designer y luego a productor y ahora director. En realidad es a medida que va creciendo el estudio, ir acomodando la cantidad de tareas que vamos teniendo y que se necesitan cubrir. Soy como un comodín.

**Respecto al prototipado de videojuegos, ¿Qué tan relevante crees que es eso?**

Es crucial. Es crucial el prototipado. Es más, el proof of concept que muchas veces terminas buscando la herramienta que siempre te queda o muy grande o muy chica y en el medio de todo esto. Vamos a decir que ya me adelantaste la idea, tener una idea directamente dedicada al prototipado es un golazo. Porque en la producción, en realidad, el prototipado viene más o de compañías muy grandes donde tienen equipos para poder prototipar cosas o las game jams y todo ese tipo de cosas que trabajas más en un prototipo que después terminas haciendo juego. Los estudios indie tenemos que tener muy claro lo que vamos hacer porque no tenemos plata ni recursos para dedicarle al prototipado. A veces, una idea puede llevarte mucho más tiempo lidiando con los motores de videojuegos que con lo que terminas trabajando. El prototipado es crucial para el estudio independiente que tiene esa idea que capaz le surgió de una game jam, que esa idea que está en la cabeza el 90% de las veces queda mal. Funcionaba bien en la cabeza, pero en los papeles termina siendo malo. En la producción, ese lapso de prototipado no suele ser el momento donde tenes plata de algún publisher o de alguien. Lo estas haciendo a pulmón. Es

super critico porque estamos en una industria donde tener a los estudios gigantes que no te quieren meter una idea nueva y van haciendo el remake del juego que ya funcionó, y una nueva ip cada cinco remakes. Y los indies que se la juegan con alguna idea que después terminan copiando. Como el fornite que en realidad nació como un indie de Day Z, que eso fue un mod. Ahora más modernos tenes a Balatro, Vampire Survivors, que salen desde un indie con una idea que yo creo que en este momento de esta crisis que estamos pasando en la industria, los que vamos a salir bien somos los estudios independientes. Y cuanto más fácil la tengamos, creo que van a salir cosas mucho mejor.

**¿Te ha pasado con algún proyecto en Bacord de que planifican tiempo para un proyecto, pero en realidad se les va de tiempo y recursos?**

Viste que todos tenemos un cajón en la cocina, en algún mueble. Ese último cajón que está lleno de cosas que no sabes dónde meter en otro lado. Estamos así pero con proyectos. Terminamos prototipando ideas, y empezar a buscar publisher y todo eso para ver si conseguimos la financiación para producirlo. Y termina quedando en un cajoncito. Proyectos siempre hay, es más, dentro del estudio yo estoy metido con un montón de gente como Lucas. A él le conseguimos que su juego, casi prototipo que fue ConurLife termine en un publisher como MicroProse. La planificación de tres meses en este juego se fue a ocho, porque al publisher le gustó el proyecto y estaba buscando mejorar la calidad. En realidad, yo creo que ponerle fecha a un prototipo es contraproducente porque estás sesgando tu capacidad para iterar. En los prototipos vos necesitas iterar, tener esos cambios que, a veces, son tan mínimos y tan fáciles pero después termina siendo algo crítico algo del juego. O ese bug que termina siendo feature.

Así que si, es muy normal que los prototipos se te vayan. Primero, siempre se van mucho más largos. Y creo que muchos prototipos terminan quedando en un cajón por que no se les pudo dar la cantidad de tiempo que estabas necesitando para que ese prototipo termine siendo un juego. Como te digo, esto de iterar e iterar hasta que te queda el producto, cuanto más pulido esté ese prototipo, de ese prototipo podemos ir a los publisher ya con algo muy cercano a un juego, como un vertical slice. Lo ideal sería prototipo, vertical slice, y ahí sale a buscar publisher y conseguir la financiación para que te produzcan el juego. La parte del prototipo siempre queda en manos del estudio, del grupo que lo esté haciendo. Es muy crítico. Creo que a la parte de prototipado se

la subestima mucho en base a la ansiedad de ver el juego final. Con Maki (juego de Bacord Games) el juego se ve muy estético pero nos quedó muy corta la parte de la jugabilidad. Y si bien ahora lo estamos tratando de arreglar. Primero hicimos el prototipo y se sentía mucho mejor que lo que terminamos haciendo en el juego, y no tuvimos mucho tiempo para iterar porque lo hicimos en seis meses. Ahí ya teníamos el contrato, ya tenes el deadline encima y la libertad que te corta esto de tener el contrato encima. Entonces, se hizo en 6 meses, se entregó y ahora estamos volviéndonos locos para mejorar la jugabilidad. Pero si hubiéramos trabajado bien, ese prototipo lo hubiéramos hecho ya pensando en la escalabilidad, no hubiéramos tenido el problema ese. Si funciona bien tu prototipo, tu juego final va a funcionar. Creo que es más fácil equivocarse en el prototipo y luego que salga mal el juego que equivocarse al principio y que salga bien. Si ese prototipo no te gusta, no funciona, no creo que tu juego final vaya a estar mucho mejor.

**¿Qué opiniones tenes respecto de la inteligencia artificial aplicada al desarrollo de videojuegos?**

El futuro va para allá, estamos frente a un cambio de paradigma. Aca en mi computadora tengo versiones locales de ChatGPT y DeepSeek. Por ejemplo, para un profesorado de inglés que estoy haciendo, nos ahorra por lo menos una ahora en la transcripción de las fotos. Después, en la parte creativa, yo suelo conversar con ChatGPT respecto a una idea que tengo, que me ayuda a darle forma. Ayuda mucho DeepSeek, ya que es mucho más creativa. Decime nombres de personajes, por ejemplo. Eso te va desatando más ideas y concatenando todo. Si tenes una IA que le digas, quiero que mi personaje haga esto, pero bien armada. Quiero que sucedan estos hechos, y poder armar el prototipo desde la IA, sería un golazo. Es más, si vos podes llegar a hacer un motor de videojuegos ya directamente para hacer eso, democratizaría mucho el tema del desarrollo porque podría venir yo que no se programar, y que tengo que estar hablando con los programadores. Y decir bueno, yo te armo el prototipo acá y después se lo paso a los desarrolladores que saben bien del tema y lo terminen escalando a un juego de en serio. Aparte, a mi me daría la posibilidad de iterar, tener tres o cuatro proof of concept en poco tiempo y me daría mucha más libertad creativa el hecho de tener yo el control de la idea. Pasar la idea de lo abstracto a algo más tangible para una persona que no se dedica directamente a eso me parece un golazo.

Yo pienso en la idea y me encanta, porque me da mucha más libertad de poder iterar, iterar sin tener costos. Si tenes un motor de videojuegos en donde podes iterar una idea y pasar de un proof of concept a un demo, un vertical slice, es un golazo.

El proof of concept tenes la idea que a veces la podes hacer en lapiz, según lo que estés haciendo. De ahí después empieza el prototipo, pero primero tenes que tener esa prueba de concepto que esté funcionando. Prototipo, luego vertical slice y luego producción. Entonces, la parte primera en donde vos vas a iterar esa idea que se te ocurrió, que tenes anotada en un papel, que la hablaste con un amigo, esa herramienta puede llegar a ayudar un montón a que estas ideas diferentes se peguen en un mercado que está saturado con la misma idea, y cada tanto tenes uno que da vuelta las cosas y dice, ahora Balatro, ahora Vampire Survivors, ahora el Fortnite.

**Si existiese una aplicación, un motor de videojuegos que integre inteligencia artificial generativa, te genere código, texturas, sprites, niveles, ¿La usarías, la adoptaría tu empresa?**

Con los ojos cerrados, es más, mientras esa herramienta ya de movida puedas decir, bueno, de acá lo voy a llevar a un motor de videojuegos en donde se pueda escalar un juego completo, es ese punto crítico que vas a tener. Porque es ese punto crítico que tenemos a la hora de hacer un proof of concept. Después, la escalabilidad que le vas dando a ese proyecto y lo llevas al juego completo, es donde más se complica, y donde más posibilidades de cometer errores tenes. Y, en esa parte de la producción, si no tenes una empresa que te de el tiempo para trabajar tranquilo, cuando ya tenes el demo armado y estas dependiendo de las fechas límite que te ponen los publishers o lo que te quieran ajustar los publishers, ya poder trabajarlo desde el concepto eso, me parece una cosa genial. Aparte, se que hay muchos detractores de la inteligencia artificial, pero yo cuando estuve hablando con los artistas, que creo que fueron los que más se le pusieron en contra, la crítica que tenían era que la inteligencia artificial está copiando el trabajo de otra persona. Pero si yo me pongo hoy a estudiar arte, primero voy a estar copiando el hombre de vitruvio del Da Vinci. Es el proceso de aprendizaje del ser humano. Luego, por otro lado está la parte moral.

Pero para los estudios chicos, nos da mucho la posibilidad de escalar más grande, de pensar más grande. Porque con una inteligencia artificial bien utilizada, el nivel de producción de un juego lo podemos elevar, porque en menos cantidad de tiempo podes hacer muchas más cosas.

**Si existiese esta herramienta, ¿qué modelo de negocios te gustaría?**

Ese es un tema peliagudo, más allá de lo que vos quieras hacer, es la competencia que tengas. Fíjate que es lo que pasó con GameMaker. Que pasaron de ese modelo de pago a un modelo de subscripción que me pareció malísimo. Pero hay gente que le es más accesible ese sistema de subscripción. Pero también tenes la competencia de Godot, que es gratis.

Es complicado este tema, porque el mercado o te puede amar la idea o te la puede desechar.

# ANEXO B: FORMULARIO DE ENCUESTAS Y RESPUESTAS

A continuación, se adjuntan imágenes del formulario de encuesta creado junto a las respuestas obtenidas. Se puede acceder al formulario a través del siguiente enlace: https://forms.gle/naT37fgWXMvBahjL7

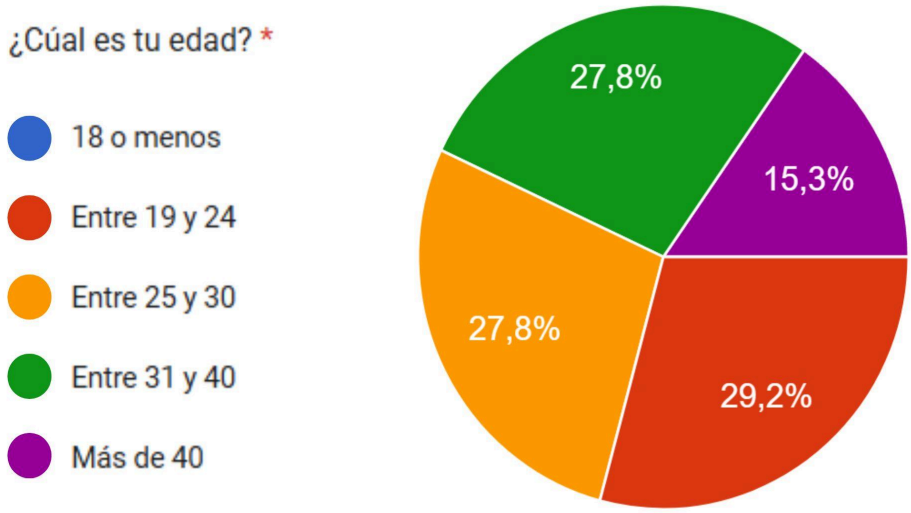


Figura 47: Pregunta "¿Cúal es tu edad?" y gráfico de respuestas. (Gráfico de elaboración propia, 2025).

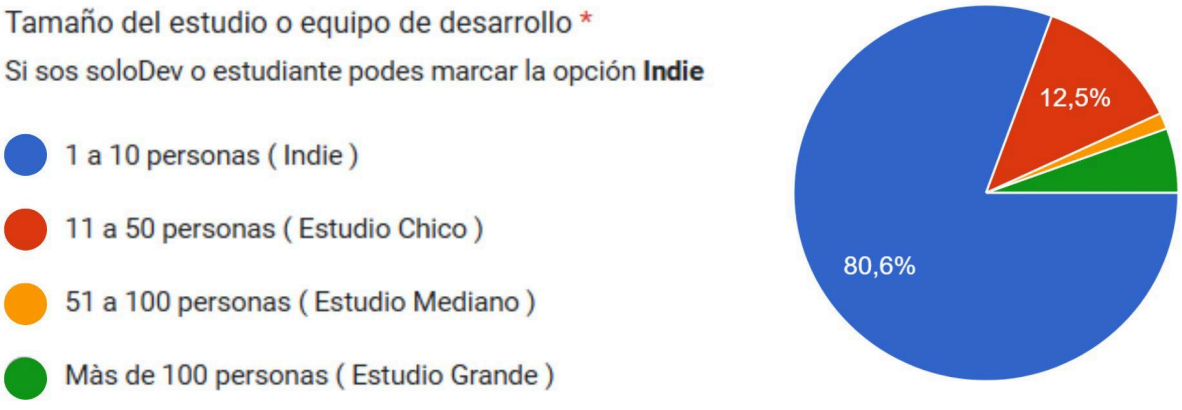


Figura 48: Consulta "Tamaño del estudio o equipo de desarrollo" y gráfico de respuestas. (Gráfico de elaboración propia, 2025).


Figura 49: Pregunta "¿Qué tan de acuerdo estás con la siguiente afirmación? *La etapa de **Prototipado** influye significativamente a la hora de desarrollar un videojuego*" y gráfico de respuestas. (Gráfico de elaboración propia, 2025).


Figura 50: Pregunta "¿Qué tan de acuerdo estás con la siguiente afirmación? *El prototipado de un videojuego puede tomar **más tiempo** del esperado*" y gráfico de respuestas. (Gráfico de elaboración propia, 2025).

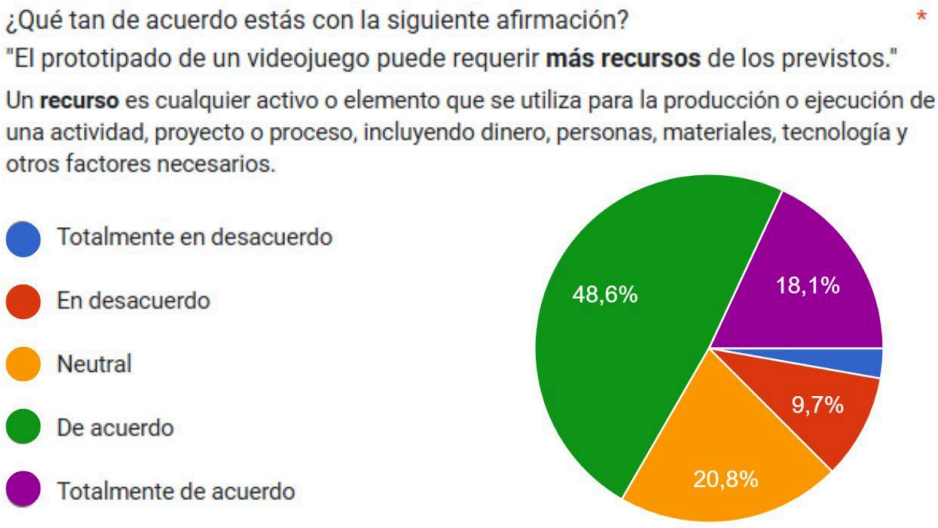


Figura 51: Pregunta "Qué tan de acuerdo estás con la siguiente afirmación? *El prototipado de un videojuego puede requerir* ***más recursos*** *de los previstos*" y gráfico de respuestas. (Gráfico de elaboración propia, 2025).

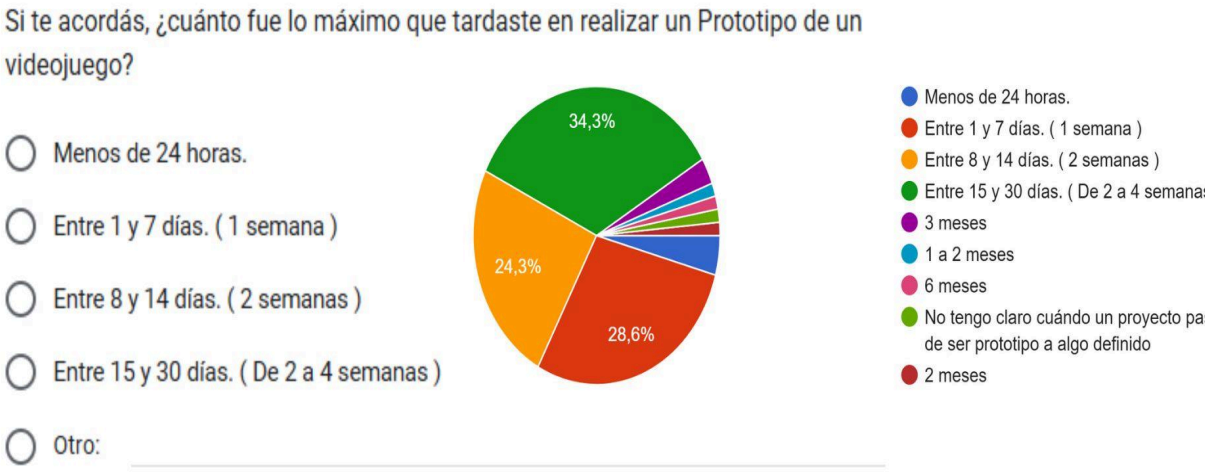


Figura 52: Pregunta "Si te acordás, ¿cuánto fue lo máximo que tardaste en realizar un Prototipo de un videojuego?" y gráfico de respuestas. (Gráfico de elaboración propia, 2025).

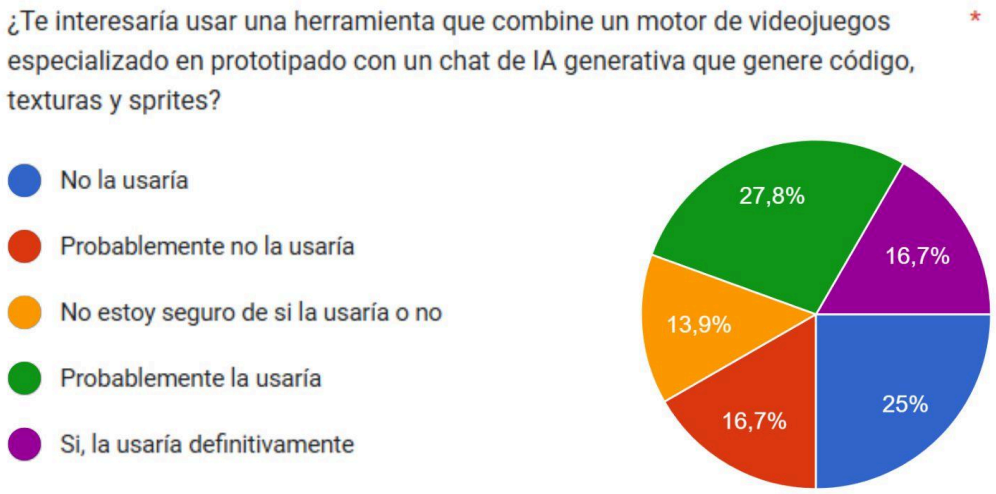


Figura 53: Pregunta "¿Te interesaría usar una herramienta que combine un motor de videojuegos especializado en prototipado con un chat de IA generativa que genere código, texturas y sprites?" y gráfico de respuestas. (Gráfico de elaboración propia, 2025).

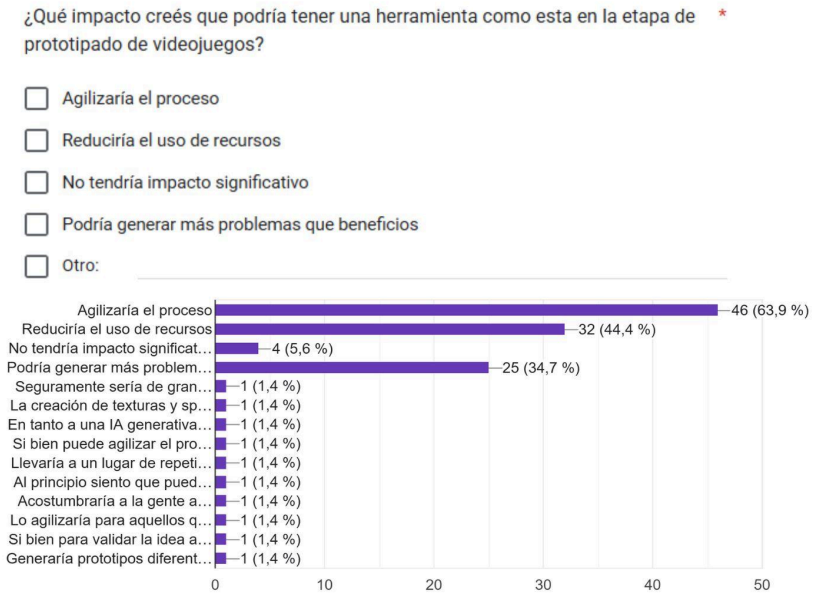


Figura 54: Pregunta de opción múltiple "¿Qué impacto creés que podría tener una herramienta como esta en la etapa de prototipado de videojuegos?"y gráfico de respuestas. (Gráfico de elaboración propia, 2025).

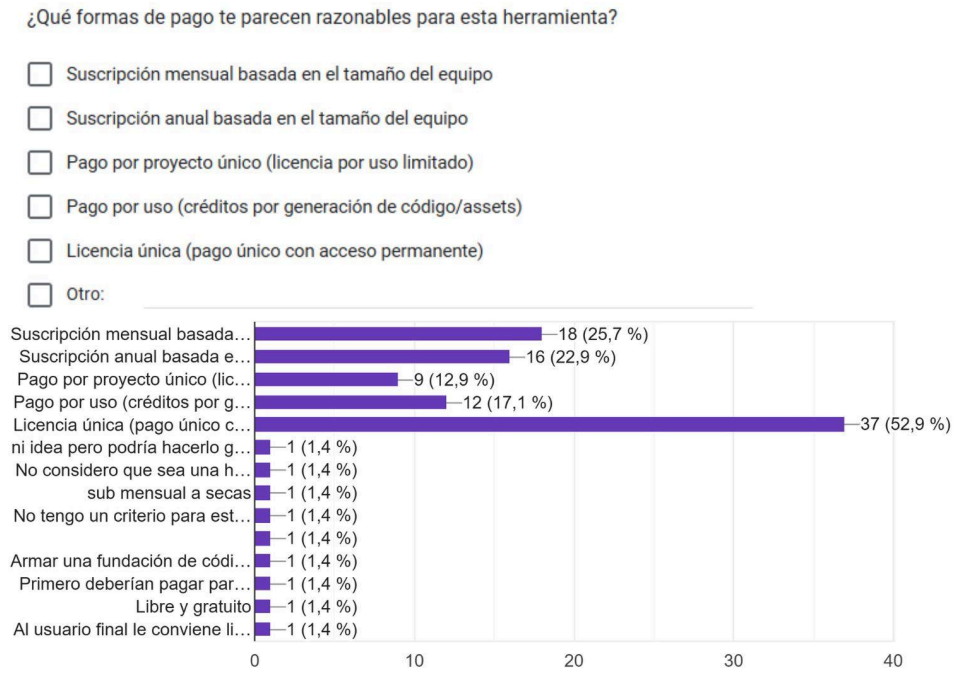


Figura 55: Pregunta de opción múltiple "¿Qué formas de pago te parecen razonables para esta herramienta?"y gráfico de respuestas. (Gráfico de elaboración propia, 2025).

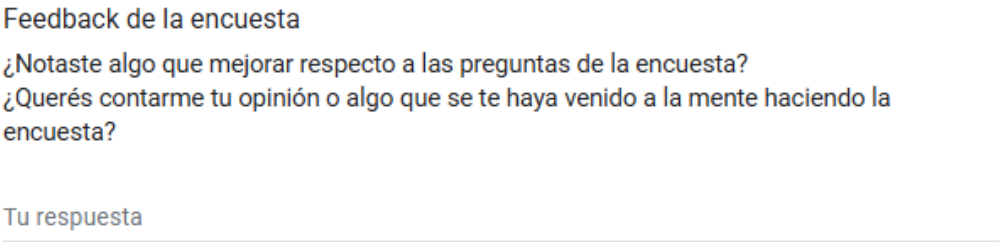


Figura 56: Pregunta de respuesta larga en busca de retroalimentación. (Gráfico de elaboración propia, 2025).

# ANEXO C: FLUJO DE FONDOS DE GAME PROTOGEN POR ESCENARIO

| Concepto | Comentarios | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| Horas desarrollador full-stack | $1,66 la hora x 300hs PFI | -$498,00 | | | | | |
| | Desarrollo de nuevas features $1.66 * 20 hs * 52 semanas en un año | | -$1.726,40 | -$1.726,40 | -$1.726,40 | -$1.726,40 | -$1.726,40 |
| Computador de Trabajo | Dell Latitude 3640 | -$600,00 | | | | | |
| Open AI | $20 Tokens para realizar pruebas en el desarrollo | -$20,00 | | | | | |
| gpt-5-mini tokens | (2M tokens input) $0.5 + (2M tokens output) + $4 * usuario | | -$562,50 | -$625,50 | -$697,50 | -$774,00 | -$864,00 |
| gpt-image-1 | (1M Tokens) $5 * cliente | | -$625,00 | -$695,00 | -$775,00 | -$860,00 | -$960,00 |
| Azure | Arquitectura para despliegue inicial: $41 * 4 meses | -$164,00 | | | | | |
| | Arquitectura Despliegue: $41 * 12 meses | | -$492,00 | -$492,00 | -$492,00 | -$492,00 | -$492,00 |
| Socio Micro ADVA | Pago anual $430,000 ARS = $301,81 | -$301,81 | -$301,81 | -$301,81 | -$301,81 | -$301,81 | -$301,81 |
| Game Jam Game Protogen | $500 en premios | | -$500,00 | -$500,00 | -$500,00 | -$500,00 | -$500,00 |
| Google Ads | USD 3/día → ≈ 6 clics diarios → ≈ 180 clics/mes → gasto anual ≈ USD 1.095. | | -$1.095,00 | -$1.095,00 | -$1.095,00 | -$1.095,00 | -$1.095,00 |
| Usuarios | Individual, Crecimiento del 10% anual por ads | | $2.200,00 | $2.420,00 | $2.660,00 | $2.920,00 | $3.220,00 |
| | Equipo, Crecimiento del 20% anual por ads | | $270,00 | $324,00 | $396,00 | $468,00 | $558,00 |
| Ingresos | | $0,00 | $2.471,00 | $2.746,00 | $3.059,00 | $3.392,00 | $3.783,00 |
| Egresos | | -$1.583,81 | -$5.302,71 | -$5.435,71 | -$5.587,71 | -$5.749,21 | -$5.939,21 |
| Flujo de Fondos NETO | | **-$1.583,81** | **-$2.831,71** | **-$2.689,71** | **-$2.528,71** | **-$2.357,21** | **-$2.156,21** |
| | | | | | | | |
| VAN | -$10.294,56 | | | | | | |
| TIR | #NUM! | | | | | | |

Figura 57: Flujo de fondos del escenario pesimista. (Gráfico de elaboración propia, 2025).

| Concepto | Comentarios | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| Horas desarrollador full-stack | $1,66 la hora x 300hs PFI | -$498,00 | | | | | |
| | Desarrollo de nuevas features $1.66 * 20 hs * 52 semanas en un año | | -$1.726,40 | -$1.726,40 | -$1.726,40 | -$1.726,40 | -$1.726,40 |
| Computador de Trabajo | Dell Latitude 3640 | -$600,00 | | | | | |
| Open AI | $20 Tokens para realizar pruebas en el desarrollo | -$20,00 | | | | | |
| gpt-5-mini tokens | (2M tokens input) $0.5 + (2M tokens output) + $4 * usuario | | -$562,50 | -$1.471,50 | -$2.052,00 | -$2.866,50 | -$4.000,50 |
| gpt-image-1 | (1M Tokens) $5 * cliente | | -$625,00 | -$1.635,00 | -$2.280,00 | -$3.185,00 | -$4.445,00 |
| Azure | Arquitectura para despliegue inicial: $41 * 4 meses | -$164,00 | | | | | |
| | Arquitectura Despliegue: $41 * 12 meses | | -$492,00 | -$492,00 | -$492,00 | -$492,00 | -$492,00 |
| Socio Micro ADVA | Pago anual $430,000 ARS = $301,81 | -$301,81 | -$301,81 | -$301,81 | -$301,81 | -$301,81 | -$301,81 |
| Game Jam Game Protogen | $500 en premios | | -$500,00 | -$500,00 | -$500,00 | -$500,00 | -$500,00 |
| Google Ads | USD 3/día → ≈ 6 clics diarios → ≈ 180 clics/mes → gasto anual ≈ USD 1.095. | | -$1.095,00 | -$1.095,00 | -$1.095,00 | -$1.095,00 | -$1.095,00 |
| Usuarios | Individual, Crecimiento del 40% anual por ads | | $4.280,00 | $6.000,00 | $8.400,00 | $11.760,00 | $16.460,00 |
| | Equipo, Crecimiento del 35% anual por ads | | $360,00 | $486,00 | $648,00 | $882,00 | $1.188,00 |
| Ingresos | | $0,00 | $4.641,00 | $6.488,00 | $9.051,00 | $12.646,00 | $17.653,00 |
| Egresos | | -$1.583,81 | -$5.302,71 | -$7.221,71 | -$8.447,21 | -$10.166,71 | -$12.560,71 |
| Flujo de Fondos NETO | | **-$1.583,81** | **-$661,71** | **-$733,71** | **$603,79** | **$2.479,29** | **$5.092,29** |
| | | | | | | | |
| VAN | $2.345,51 | | | | | | |
| TIR | 30% | | | | | | |

Figura 58: Flujo de fondos del escenario neutro. (Gráfico de elaboración propia, 2025).

| Concepto | Comentarios | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| Horas desarrollador full-stack | $1,66 la hora x 300hs PFI | -$498,00 | | | | | |
| | Desarrollo de nuevas features $1.66 * 40 hs * 52 semanas en un año | | -$1.726,40 | -$1.726,40 | -$1.726,40 | -$1.726,40 | -$1.726,40 |
| Computador de Trabajo | Dell Latitude 3640 | -$600,00 | | | | | |
| Open AI | $20 Tokens para realizar pruebas en el desarrollo | -$20,00 | | | | | |
| gpt-5-mini tokens | (2M tokens input) $0.5 + (2M tokens output) + $4 * usuario | | -$562,50 | -$1.966,50 | -$2.943,00 | -$4.401,00 | -$6.588,00 |
| gpt-image-1 | (1M Tokens) $5 * cliente | | -$625,00 | -$2.185,00 | -$3.270,00 | -$4.890,00 | -$7.320,00 |
| Azure | Arquitectura para despliegue inicial: $41 * 4 meses | -$164,00 | | | | | |
| | Arquitectura Despliegue: $41 * 12 meses | | -$492,00 | -$492,00 | -$492,00 | -$492,00 | -$492,00 |
| Socio Micro ADVA | Pago anual $430,000 ARS = $301,81 | -$301,81 | -$301,81 | -$301,81 | -$301,81 | -$301,81 | -$301,81 |
| Game Jam Game Protogen | $500 en premios | | -$500,00 | -$500,00 | -$500,00 | -$500,00 | -$500,00 |
| Google Ads | USD 3/día → ≈ 6 clics diarios → ≈ 180 clics/mes → gasto anual ≈ USD 1.095. | | -$1.095,00 | -$1.095,00 | -$1.095,00 | -$1.095,00 | -$1.095,00 |
| Usuarios | Individual, Crecimiento del 50% anual por ads | | $5.340,00 | $8.020,00 | $12.040,00 | $18.060,00 | $27.100,00 |
| | Equipo, Crecimiento del 45% anual por ads | | $450,00 | $648,00 | $936,00 | $1.350,00 | $1.962,00 |
| Ingresos | | $0,00 | $5.791,00 | $8.670,00 | $12.979,00 | $19.414,00 | $29.067,00 |
| Egresos | | -$1.583,81 | -$5.302,71 | -$8.266,71 | -$10.328,21 | -$13.406,21 | -$18.023,21 |
| Flujo de Fondos NETO | | **-$1.583,81** | **$488,29** | **$403,29** | **$2.650,79** | **$6.007,79** | **$11.043,79** |
| | | | | | | | |
| VAN | $11.203,53 | | | | | | |
| TIR | 93% | | | | | | |

Figura 59: Flujo de fondos del escenario optimista. (Gráfico de elaboración propia, 2025).