# Arquitectura Técnica y Especificaciones Sistémicas para el Desarrollo de un Simulador Web del Juego de Cartas 26

El presente reporte detalla la investigación técnica y el diseño de sistemas para la digitalización del juego de cartas "26", una herramienta lúdico-pedagógica desarrollada por el Observatorio del Juego. Esta obra se fundamenta en la necesidad de transformar una experiencia física de alta carga cognitiva y matemática en un entorno virtual síncrono para dos jugadores, garantizando la integridad de las reglas, la fluidez de la comunicación en tiempo real y la robustez de la lógica de validación. La transición del formato físico al digital no es una mera transposición de imágenes, sino una reingeniería de procesos que requiere la formalización de algoritmos de secuenciación, la gestión de estados compartidos y la implementación de mecanismos de seguridad para evitar la manipulación de resultados.

## Contextualización y Propósito Pedagógico del Juego 26

El juego "26" se sitúa en la intersección de la gamificación educativa y el desarrollo de habilidades lógico-matemáticas. Según las directrices del Observatorio del Juego, esta actividad está diseñada para jugadores mayores de 8 años, con un nivel de complejidad media y una duración estimada de entre 60 y 90 minutos.1 A diferencia de otros juegos de cartas que dependen del azar o de bazas simples, "26" exige una planificación estratégica a largo plazo y una gestión eficiente de recursos limitados.

El propósito fundamental del simulador es facilitar la práctica de la resolución de problemas en un entorno controlado. El juego original utiliza el naipe inglés para trabajar la red numérica y la capacidad de anticipación.3 En su versión física, el juego permite que los participantes se familiaricen con la estructura de las escalas y el valor relativo de las cartas, mientras que la versión digital debe añadir una capa de análisis de datos que permita a docentes o investigadores observar el proceso de toma de decisiones de los jugadores.

## Análisis Exhaustivo de las Reglas y Mecánicas de Juego

La digitalización del juego requiere una comprensión absoluta de sus tres pilares: la distribución, la ejecución de turnos y las condiciones de victoria. El juego se basa en el naipe inglés (52 cartas) e incorpora 12 comodines especiales que alteran la linealidad de las escalas.1

### Distribución Inicial y Componentes del Mazo

Para una partida de dos jugadores, el simulador debe gestionar un total de 52 cartas más los comodines. La distribución es asimétrica entre las zonas de juego de cada participante, lo que introduce un elemento de gestión de inventario crítico.

| **Componente de Juego** | **Cantidad por Jugador** | **Visibilidad** | **Función Técnica** |
| --- | --- | --- | --- |
| Mano Activa | 6 cartas | Privada (solo el jugador) | Recurso inmediato para jugar en escalas o descartar.1 |
| Pila Inicial (Reserva) | 20 cartas | Mixta (19 cerradas, 1 abierta) | Principal obstáculo de victoria; motor de flujo de juego.1 |
| Comodines Reservados | Variable (máximo 12 total) | Pública | Cartas K y Jokers que se extraen de la mano al recibirlas.1 |
| Mazo Central (Stock) | Cartas restantes | Oculta | Fuente de reposición al inicio de cada turno. |

El sistema de distribución debe asegurar que cada jugador comience con exactamente 26 cartas de la baraja asignadas a sus pilas y manos iniciales, lo que justifica el nombre del juego.1 La pila de 20 cartas es el núcleo de la estrategia: el jugador debe priorizar jugar la carta que se encuentra en la parte superior de esta pila para poder revelar la siguiente y avanzar hacia el vaciado total.1

### Dinámica de las Escalas y el Centro de la Mesa

El centro de la mesa es el espacio de juego compartido donde se pueden formar hasta 4 escalas simultáneamente.1 Una escala es una secuencia estrictamente ascendente que comienza obligatoriamente con un As (valor 1) y culmina en una Reina o Q (valor 12).1

La lógica de construcción de escalas posee las siguientes restricciones que el simulador debe validar en cada movimiento:

* Las escalas son independientes de la pinta o palo de la carta; solo importa el valor numérico.1
* Cualquier jugador puede contribuir a cualquiera de las 4 escalas activas.
* Si se intenta iniciar una quinta escala, el sistema debe bloquear el movimiento a menos que una de las anteriores se haya completado y retirado.
* Una vez que una escala alcanza la Q, se considera cerrada y el espacio queda disponible para una nueva secuencia que inicie con un As.1

### El Rol de los Comodines y Reglas de Adyacencia

El juego 26 incorpora doce comodines: 4 Jokers y 8 cartas de Rey (K).1 La función del comodín es sustituir cualquier valor numérico dentro de la escala, con dos excepciones críticas: el As y la Reina. Esto significa que un comodín nunca puede ser la primera ni la última carta de una escala.1

Desde la perspectiva del modelado de datos, la gestión de comodines introduce una regla de no concurrencia: no se pueden colocar dos comodines de forma consecutiva en la misma escala.1 Debe existir al menos una carta de valor natural entre ellos. Esto obliga al sistema a mantener un historial del tipo de carta (natural vs. comodín) para cada posición de la escala.

### Las Columnas de Descarte Personal

Cada jugador dispone de hasta 4 columnas de descarte personal. Estas columnas sirven como almacenamiento de largo plazo para cartas que no pueden jugarse inmediatamente en las escalas centrales.1 La regla física establece que las cartas de mayor valor (cercanas a Q) deben estar más ocultas que las de menor valor (cercanas al As). En el simulador, esto se traduce en una estructura de datos tipo Pila (Stack) donde solo el elemento superior es plenamente accesible para ser jugado de vuelta al centro de la mesa, incentivando al jugador a planificar el orden en que se deshace de sus cartas.1

## Lógica de Validación Matemática y Reglas de Negocio

El simulador debe actuar como un árbitro imparcial que impone las reglas mediante un motor de validación. Este motor no solo comprueba la legalidad de un movimiento, sino que también gestiona las transiciones de estado automáticas.

### El Algoritmo de Validación de Escalas

Para cada intento de colocar una carta  en una escala , el algoritmo debe ejecutar la siguiente secuencia lógica:

1. Si  está vacía: Validar que  sea un As.
2. Si  contiene cartas :
   * Si  es un comodín: Validar que  no sea un comodín y que la posición de  no sea la última de la secuencia (posición 12).1
   * Si  es una carta natural de valor : Validar que  sea exactamente el sucesor del valor representado por . Si  era un comodín, el valor representado por  se infiere del valor de .

### Gestión de la Mano y el Robo Automático

Al inicio de cada turno, el jugador activo debe tener 6 cartas en su mano. El simulador debe realizar una operación de llenado desde el mazo central.1 Durante este proceso, si el sistema detecta que una de las cartas robadas es un comodín (K o Joker), debe ejecutar una acción secundaria automática:

* Mover el comodín a la reserva de comodines del jugador.
* Robar una nueva carta del mazo central para que el recuento de cartas en mano se mantenga en 6.1

Este comportamiento es una de las mayores ventajas del simulador digital, ya que elimina el error humano de olvidar reservar los comodines o de no reponer la mano correctamente.

### El Ciclo de Fin de Turno

Un turno no termina hasta que el jugador realiza un descarte obligatorio en una de sus 4 columnas personales.1 Esta acción es irrevocable y transfiere el control del estado "jugador activo" al oponente. El sistema debe impedir el descarte si el jugador aún tiene movimientos posibles con la carta superior de su pila de 20, aunque las reglas permiten al jugador elegir estratégicamente cuándo dejar de jugar y descartar para pasar el turno.

### La Variante "Quinta": Matemáticas Aplicadas

Investigaciones adicionales del Ministerio de Educación de Chile y guías pedagógicas sugieren variantes del juego de cartas donde el número "26" es el objetivo de una construcción aritmética.5 En la modalidad "Quinta", se reparten 5 cartas a cada jugador y se generan dos cartas centrales que forman un número de dos dígitos (ej. un 2 y un 6 forman el 26).5

El objetivo en esta variante es utilizar las 5 cartas de la mano mediante operaciones básicas (suma, resta, multiplicación y división) para alcanzar exactamente el número objetivo.6 Para el simulador web, esto implica la creación de un evaluador de expresiones matemáticas.

| **Característica** | **Regla de Validación** | **Implicación Técnica** |
| --- | --- | --- |
| Uso de Operandos | Se deben usar las 5 cartas obligatoriamente en la versión avanzada.6 | Validación de uso de todos los nodos del árbol de expresión. |
| Operadores Permitidos | , , , | Implementación de precedencia de operadores (PEMDAS).7 |
| Tiempo Límite | Generalmente 2 o 5 minutos por ronda.5 | Implementación de setInterval síncrono vía WebSockets. |
| Puntuación | 1 punto por lograr el número exacto.5 | Actualización de tabla de líderes en tiempo real. |

El simulador debe integrar esta modalidad como un "Modo de Desafío" adicional, utilizando un parser de expresiones que evite el uso de eval() por razones de seguridad, prefiriendo bibliotecas de análisis sintáctico que construyan un Árbol de Sintaxis Abstracta (AST) para evaluar la validez de la operación propuesta por el jugador.8

## Modelado de Datos y Arquitectura de Persistencia

La creación de un simulador robusto requiere un esquema de base de datos que refleje fielmente la complejidad de una partida de cartas. La utilización de Supabase, basada en PostgreSQL, permite manejar la integridad relacional y los cambios en tiempo real.11

### Esquema de Base de Datos (PostgreSQL)

Se proponen tres tablas principales para gestionar el ciclo de vida de una partida de dos jugadores:

1. **games**: Almacena el estado global de la partida.
   * id (uuid, PK)
   * status (enum: 'waiting', 'active', 'finished')
   * current\_turn (uuid, FK a players)
   * stock\_deck (jsonb: arreglo de cartas restantes)
   * central\_scales (jsonb: arreglo de 4 arreglos de cartas)
   * created\_at, updated\_at
2. **players**: Detalles de los participantes en cada partida.
   * id (uuid, PK)
   * game\_id (uuid, FK)
   * user\_id (uuid, FK a auth.users)
   * hand (jsonb: cartas en mano)
   * main\_pile (jsonb: cartas en la pila de 20)
   * wildcard\_reserve (jsonb: comodines apartados)
   * discard\_columns (jsonb: 4 pilas de descarte)
   * is\_ready (boolean)
3. **moves\_log**: Historial de acciones para auditoría y funciones de "deshacer" o repetición.
   * id (uuid, PK)
   * player\_id (uuid, FK)
   * action\_type (text: 'play\_to\_scale', 'discard', 'draw')
   * payload (jsonb: detalle del movimiento)

### El Estado de la Carta como Objeto Atómico

Cada carta en el sistema debe representarse como un objeto con propiedades inmutables para facilitar la depuración y la renderización en el frontend:

JSON

{  
 "id": "c102-h5",  
 "rank": 5,  
 "suit": "hearts",  
 "value": 5,  
 "is\_wildcard": false,  
 "metadata": {  
 "is\_face\_up": true,  
 "owner\_id": "user-abc"  
 }  
}

La propiedad id es crucial para las animaciones de transición (Shared Element Transitions) en el navegador, permitiendo que la misma carta parezca viajar físicamente de la mano de un jugador a una escala central.12

## Arquitectura de Comunicación en Tiempo Real

Para un juego de dos jugadores, la sincronización es el componente técnico más crítico. La latencia entre que un jugador coloca una carta y el otro la ve puede arruinar la experiencia competitiva.

### Sincronización mediante Canales de Supabase (Realtime)

La arquitectura debe aprovechar las tres funcionalidades clave de Supabase Realtime para diferentes propósitos 13:

* **Broadcast:** Se utiliza para enviar eventos efímeros que no requieren persistencia estricta, como la posición del cursor del oponente o la animación de "arrastrando carta". Esto mejora la sensación de presencia sin sobrecargar la base de datos.13
* **Presence:** Crucial para el lobby y la detección de desconexiones. Si un jugador abandona la pestaña, el sistema lo detecta inmediatamente mediante el "heartbeat" del socket y puede pausar la partida o declarar una victoria por abandono.13
* **Postgres Changes:** Es la fuente de verdad. Cuando una acción es validada y guardada en la tabla games o players, Supabase notifica a ambos clientes para que actualicen su estado local (Zustand o Redux). Esto garantiza que ambos jugadores vean siempre la misma configuración de escalas y pilas.11

### El Bucle de Juego (Game Loop) y Matchmaking

El flujo de inicio de una partida sigue este proceso:

1. **Creación de Sala:** El jugador 1 crea una entrada en games y recibe un código de acceso.
2. **Unión de Jugador:** El jugador 2 ingresa el código y se registra en la tabla players vinculada a ese game\_id.14
3. **Sincronización de Inicio:** Cuando ambos envían is\_ready: true, el servidor ejecuta una Función de Borde (Edge Function) que genera el mazo, lo baraja usando el algoritmo de Fisher-Yates y distribuye las cartas iniciales en la base de datos.12
4. **Notificación de Turno:** El sistema designa al primer jugador (según las reglas, el de la derecha del repartidor en el formato físico, o aleatoriamente en el digital) y activa su capacidad de interacción en la UI.1

## Interfaz de Usuario y Experiencia de Usuario (UI/UX)

La interfaz del simulador debe ser intuitiva, minimizando la carga cognitiva para que el jugador se concentre en el desafío matemático.

### Diseño del Tablero Virtual

Se recomienda una disposición espacial que imite una mesa de juego real pero optimizada para pantallas:

* **Centro:** Las 4 escalas centrales, con indicadores de "Próximo número necesitado".
* **Zona Inferior (Jugador Local):** Mano de 6 cartas con efecto de abanico, pila de 20 cartas a la izquierda, reserva de comodines y las 4 columnas de descarte de fácil acceso.
* **Zona Superior (Oponente):** Versión espejada pero con cartas de mano ocultas. La pila de 20 del oponente debe mostrar claramente cuántas cartas le quedan para ganar, fomentando la presión competitiva.1

### Interacción Drag-and-Drop y Animaciones

El uso de bibliotecas como dnd-kit en el ecosistema React/Next.js es fundamental para permitir que los jugadores arrastren cartas hacia las escalas.11

* **Feedback Visual:** Cuando una carta se arrastra sobre una escala, el borde de la escala debe iluminarse en verde si el movimiento es válido y en rojo si es inválido, según la lógica de validación matemática discutida previamente.
* **Optimismo en la UI:** El simulador debe aplicar el movimiento localmente de forma inmediata (Optimistic UI) y luego revertirlo si la base de datos responde con un error de validación, evitando la sensación de "lag".15

## Estrategia de Desarrollo Agéntico con Google Antigravity

Para la construcción de este simulador, se propone el uso de Google Antigravity, una plataforma de desarrollo agéntica que permite orquestar agentes de IA para planificar, codificar y probar aplicaciones de forma autónoma.16

### Uso de Antigravity en el Flujo de Trabajo

La implementación del simulador se puede dividir en "misiones" gestionadas desde el Administrador de Agentes (Agent Manager) de Antigravity 17:

1. **Fase de Planificación:** El desarrollador proporciona un prompt de alto nivel: "Crea un sistema de validación para escalas de cartas del As a la Q, integrando la lógica de no-concurrencia de comodines". Antigravity generará un Plan de Implementación que detalla la estructura de archivos y la lógica algorítmica.18
2. **Codificación Asíncrona:** Mientras el desarrollador se enfoca en el diseño visual en el Editor, un agente de Antigravity puede trabajar en segundo plano configurando los tipos de TypeScript para la base de datos de Supabase y generando los hooks de comunicación en tiempo real.17
3. **Pruebas en el Browser Agent:** Antigravity incluye un agente de navegador que puede ejecutar el simulador, simular clics y verificar visualmente que las cartas se muevan correctamente, capturando errores de interfaz antes de que lleguen a producción.17
4. **Despliegue y CI/CD:** Utilizando la integración nativa de Antigravity con GitHub, el proyecto se puede publicar y conectar con Netlify para un despliegue automático de la versión web, gestionando las variables de entorno de Supabase de forma segura.19

| **Herramienta Antigravity** | **Aplicación en el Proyecto 26** |
| --- | --- |
| **Agent Manager** | Supervisión de los bots que escriben la lógica de las reglas del juego.17 |
| **Protocolo de Artefactos** | Revisión del plan de red numérica antes de escribir el código de las escalas.18 |
| **Terminal Control** | Ejecución de suites de pruebas unitarias para validar el reparto equitativo de cartas.17 |

## Seguridad e Integridad del Juego

Un simulador web es vulnerable a ataques de inyección y manipulación de estado. La seguridad debe abordarse en múltiples capas.

### Validación en el Lado del Servidor (Edge Functions)

Nunca se debe confiar exclusivamente en la validación del cliente. Cada movimiento enviado debe ser re-validado por una Supabase Edge Function antes de actualizar la base de datos.12 Esta función comprueba:

* ¿Es el turno del jugador que envía el movimiento?
* ¿Posee el jugador la carta que intenta jugar en su mano o pila?
* ¿Es el movimiento válido según el estado actual de la escala central?

### Protección contra el "Cheating" y Manipulación de Mazo

El orden del mazo central y de las pilas de 20 cartas nunca debe ser enviado completo al cliente. El cliente solo debe conocer la carta superior de la pila y las cartas en su propia mano.12 El resto del mazo permanece encriptado o almacenado únicamente en el servidor, revelándose carta por carta a medida que el juego progresa. Esto evita que usuarios con conocimientos técnicos inspeccionen el estado de la aplicación para saber qué cartas vienen a continuación.

## Conclusiones y Recomendaciones para la Implementación

El desarrollo del simulador web "26" es un proyecto que demanda una alta fidelidad a las reglas del Observatorio del Juego mientras se implementan tecnologías modernas de sincronización. La investigación demuestra que la combinación de Next.js para el frontend, Supabase para el backend en tiempo real y Google Antigravity para la aceleración del desarrollo constituye el stack tecnológico óptimo.

**Recomendaciones finales:**

* **Priorizar la Pila de 20:** La interfaz debe centrar la atención del jugador en su pila de 20 cartas, ya que vaciarla es la condición necesaria para desbloquear el final de la partida.1
* **Feedback Educativo:** Integrar un modo de visualización de errores que explique por qué un movimiento no es válido, reforzando el aprendizaje de la secuencia numérica.
* **Escalabilidad:** Diseñar el modelo de datos para permitir en el futuro partidas de hasta 4 jugadores, como permite el juego físico original, aunque la fase inicial se centre en 2 jugadores.1
* **Accesibilidad:** Asegurar que los colores de las cartas y las escalas sean distinguibles para personas con daltonismo, utilizando símbolos o etiquetas de texto adicionales.

Este reporte sirve como el documento técnico base para la fase de desarrollo, asegurando que todos los involucrados tengan una visión clara de la lógica matemática, la arquitectura de datos y los flujos de trabajo necesarios para entregar un simulador de clase mundial que honre el espíritu educativo del Observatorio del Juego.

#### Obras citadas

1. ¿QUIÉN GANA EL JUEGO?, fecha de acceso: febrero 1, 2026, <https://www.cicpolmue.cl/wp-content/uploads/2020/06/JUEGO26.pdf>
2. fecha de acceso: febrero 1, 2026, <https://www.observatoriodeljuego.cl/products/juego-tip-top-26#:~:text=%2226%22%20es%20un%20juego%20de,en%20las%20columnas%20de%20descarte.>
3. SIGMA 26 - LOS JUEGOS EN MATEMÁTICAS - Euskadi.eus, fecha de acceso: febrero 1, 2026, <https://www.euskadi.eus/contenidos/informacion/dia6_sigma/es_sigma/adjuntos/sigma_26/2_juegos_matematicas.pdf>
4. Juego tip top , 26 - Observatorio del juego, fecha de acceso: febrero 1, 2026, <https://www.observatoriodeljuego.cl/products/juego-tip-top-26>
5. ¡COMENCEMOS!, fecha de acceso: febrero 1, 2026, <https://www.curriculumnacional.cl/614/articles-343688_recurso_pdf.pdf>
6. Quinta Instrucciones, fecha de acceso: febrero 1, 2026, <https://www.curriculumnacional.cl/sites/default/files/adjuntos/recursos/2025-01/Instrucciones.pdf>
7. Juegos de Cartas, fecha de acceso: febrero 1, 2026, <https://resources.finalsite.net/images/v1698218795/busdk12caus/ygmiyjmtxrkcrsol7axr/Card_DiceFluencyGamesSpanish.pdf>
8. A Safer Alternative to JavaScript's eval: Introducing safeEval | by Sai Dheeraj V | Medium, fecha de acceso: febrero 1, 2026, <https://medium.com/@saidheerajv/a-safer-alternative-to-javascripts-eval-introducing-safeeval-ecd13f3661cd>
9. alternative to eval : r/learnjavascript - Reddit, fecha de acceso: febrero 1, 2026, <https://www.reddit.com/r/learnjavascript/comments/1n8klp9/alternative_to_eval/>
10. Alternative for eval() in javascript for expression evaluation, fecha de acceso: febrero 1, 2026, <https://softwareengineering.stackexchange.com/questions/323592/alternative-for-eval-in-javascript-for-expression-evaluation>
11. Built a real-time multiplayer game with Next.js (App Router) + Zustand + Supabase — no custom backend : r/reactjs - Reddit, fecha de acceso: febrero 1, 2026, <https://www.reddit.com/r/reactjs/comments/1kyi6xn/built_a_realtime_multiplayer_game_with_nextjs_app/>
12. Building Scalable Real-Time Multiplayer Card Games - DEV Community, fecha de acceso: febrero 1, 2026, <https://dev.to/krishanvijay/building-scalable-real-time-multiplayer-card-games-3kn6>
13. Realtime: Multiplayer Edition - Supabase, fecha de acceso: febrero 1, 2026, <https://supabase.com/blog/supabase-realtime-multiplayer-general-availability>
14. Exploring Supabase Realtime By Building a Game - aleksandra.codes, fecha de acceso: febrero 1, 2026, <https://www.aleksandra.codes/supabase-game>
15. How I Built a Multiplayer Guessing Game Using React, Supabase & Next.js - Medium, fecha de acceso: febrero 1, 2026, <https://medium.com/@yogeshrana2301/how-i-built-a-multiplayer-guessing-game-using-react-supabase-next-js-a39a94c7825a>
16. ¿Qué es la antigravedad de Google? Guía completa, características, límites y ejemplos reales - Q2B Studio, fecha de acceso: febrero 1, 2026, <https://www.q2bstudio.com/nuestro-blog/317165/descubre-todo-sobre-la-antigravedad-de-google-caracteristicas-limites-ejemplos-reales-y-como-funciona-en-esta-guia-completa-conviertete-en-un-experto-en-esta-innovadora-tecnologia>
17. Google Antigravity: El Nuevo IDE de IA con Gemini 3 y Agentes - Kreativos Pro, fecha de acceso: febrero 1, 2026, <https://kreativos.pro/google-antigravity-ide-ia-gemini-3-vs-easter-egg/>
18. Primeros pasos con Google Antigravity, fecha de acceso: febrero 1, 2026, <https://codelabs.developers.google.com/getting-started-google-antigravity?hl=es-419>
19. Cómo Hacer Deploy de tu App Web desde Google Antigravity (Gratis) - Rodrigo Olivares, fecha de acceso: febrero 1, 2026, <https://rodrigoolivares.com/publicar-app-antigravity-github-netlify/>