Documento de definición - Juego e-Instant (Phaser 3 sobre React)

Versión: v3 (sin bonus, backend stub, replay por ticket en paquetes) | Fecha: 2026-01-05

Este documento define el alcance funcional y técnico del MVP del juego e-Instant, considerando integración con un backend temporal ("backend de palo") y soporte multi-cliente mediante configuración por código de cliente/empresa.

1. Objetivo

- Implementar un MVP del juego en frontend (React + Phaser) que reproduzca el flujo de juego base con cascadas.
- Permitir habilitación multi-cliente sin reprogramar el juego: cambiar clientCode/companyCode (y configuración) debe ser suficiente.
- Integrarse con un backend temporal que entregue resultados (outcomes) consistentes para animación y pruebas.
- En modo Paquete, permitir revisar/repetir cada jugada individual (replay) y volver a la lista para elegir otra.

2. Alcance del MVP

- Tres modos de ejecución: Nivel 1, Nivel 2 y Paquete de Tickets (x5/x10/x15/x20).
- Tablero tipo grilla (ej. 4x6) con evaluación por agrupaciones (clusters) y mecánica de cascadas (remove + drop + refill).
- Pantallas: selección de experiencia, loading, juego principal (play), lista de tickets en paquete, replay por ticket, y resumen del paquete.
- Configuración multi-cliente: branding, parámetros de apuesta, textos, moneda y límites, provistos por backend.

3. Fuera de alcance (por ahora)

- Bonos y mini-juegos (BONUS, free plays, level-up bonus).
- Implementación fina del backend definitivo (microservicios, auditoría completa, wallet real, pagos).
- Jackpots/acumulados reales (solo UI si se requiere, no lógica financiera).
- Cumplimiento regulatorio final (provably fair, certificación) - se deja considerado a nivel de interfaces.

4. Modos de juego

4.1 Nivel 1

Modo base de juego con una configuración de volatilidad definida (pesos de símbolos y tabla de pagos). Se ejecuta un ticket a la vez.

- Input: bet (apuesta), modo = nivel1.

- Output: secuencia de cascadas + totalWin.
- UX: Play -> reveal tablero -> cascadas -> resultado.

4.2 Nivel 2

Modo alternativo con distinta configuración de volatilidad respecto a Nivel 1. La mecánica visual es la misma; cambian parámetros (weights/paytable, etc.).

- Input: bet, modo = nivel2.
- Output: secuencia de cascadas + totalWin (esperado: distinta distribución estadística).

4.3 Paquete de Tickets (x5/x10/x15/x20)

Ejecución de N tickets como paquete. Además de mostrar la mejor jugada, el usuario puede entrar a ver la repetición (replay) de cualquier ticket del paquete y volver atrás para elegir otro.

4.3.1 Requerimientos funcionales

- Selección: packSize ∈ {5, 10, 15, 20}.
- Ejecución: se obtienen N outcomes (tickets) desde backend (en una llamada o N llamadas).
- Vista 'Lista de tickets': se muestran N items (ej. Ticket 1..N) con win por ticket (si corresponde) y un botón/ver acción 'Repetir' o 'Ver'.
- Vista 'Replay ticket': reproduce la animación completa del ticket seleccionado (reveal + cascadas).
- Navegación: desde Replay se puede volver a la lista y seleccionar otro ticket para repetir.
- La funcionalidad de replay debe funcionar offline respecto a backend (si ya se recibieron los outcomes): se reproduce desde el outcome almacenado en memoria.

4.3.2 Requerimientos de datos

- El backend debe entregar outcomes por ticket suficientemente completos para replay (grid inicial + cascades[] + totalWin).
- Identificador estable por ticket dentro del pack: packId + ticketIndex o playId.
- Opcional: payload liviano para UI en la lista (win, flags), y payload completo para replay (cascades).

5. Flujo de usuario (alto nivel)

1. Seleccionar experiencia (Nivel 1 o Nivel 2).
2. Opcional: seleccionar paquete de tickets (x5/x10/x15/x20) o desactivar.
3. Pantalla de carga (preload).
4. Pantalla principal: ajustar apuesta (bet) y presionar Play.
5. Ejecución del ticket: reveal del tablero, cascadas y cálculo de premio.
6. Si es paquete: se presenta una vista con (a) mejor jugada, (b) lista de tickets, (c) resumen del paquete.
7. El usuario puede entrar a 'Replay' de cualquier ticket y volver atrás a la lista para ver otro.

6. Mecánica de cascadas (definición lógica)

El resultado de un ticket se compone de uno o más pasos de cascada. En cada paso:

- Se evalúa la grilla para encontrar combinaciones ganadoras (ej. clusters de 3+ iguales adyacentes).
- Se eliminan las celdas ganadoras (remove).

- Se aplica gravedad por columna (drop) y se rellenan nuevas piezas arriba (refill).
- Se calcula el winStep de ese paso y se acumula al total.
- Se repite hasta que no existan nuevas combinaciones (maxCascades configurable).

7. Multi-cliente (clientCode/companyCode)

El juego debe ser multi-tenant: el mismo build de frontend debe servir a múltiples clientes/empresas solo cambiando configuración. El clientCode/companyCode se envía en cada request al backend.

7.1 Principios

- Sin forks del juego por cliente: no se recompila ni se reprograma lógica por cada empresa.
- Configuración remota: al iniciar, el frontend consume un endpoint de configuración por clientCode.
- Feature flags: habilitar/deshabilitar modos (nivel1/nivel2/pack) por cliente.
- Theming: logos, backgrounds, tipografías (si aplica), textos, moneda, límites.

7.2 Parámetros configurables (ejemplos)

- Moneda y formato: CLP/USD, separadores, decimales.
- Rangos de apuesta: minBet, maxBet, step.
- Tamaños de paquete habilitados: [5,10,15,20] o subset.
- Tamaño de grilla: rows/cols (si se decide hacerlo configurable).
- Set de símbolos y mapeo a assets (atlas).
- Pesos (weights) y tabla de pagos (paytable) por modo.

8. Arquitectura frontend (React + Phaser)

- React: UI/HUD, navegación (pantallas), selección de experiencia/paquete, lista de tickets y navegación a replay.
- Phaser: render del tablero, animación de reveal y cascadas, reproducción de replay.
- Bridge/event bus: canal de eventos ui:* y game:* para desacoplar UI y motor.
- State machine en Phaser: MENU -> LOADING -> READY -> REVEAL -> CASCADE_LOOP -> END_TICKET -> PACK_LIST/REPLAY/RESUMEN.

9. Backend temporal ("backend de palo")

El backend definitivo será implementado por otro equipo y eventualmente por microservicios. Para el MVP se requiere un backend temporal sencillo que entregue outcomes. La implementación no es crítica a nivel fino; lo importante es respetar contratos y consistencia.

9.1 Responsabilidades mínimas del backend stub

- Entregar configuración por clientCode/companyCode (theming y parámetros).
- Entregar outcome de un ticket: grilla inicial + pasos de cascada + totalWin.
- Para paquete: entregar N outcomes completos para permitir replay por ticket.

10. Contratos API propuestos (MVP)

Método	Ruta	Propósito	Request (resumen)	Response (resumen)
GET	/v1/game-config?clientCode=...&gameCode=...	Config por cliente	clientCode, gameCode	branding, betOptions, modes, packOptions, boardSpec, symbols, paytables/weights
POST	/v1/play	Ejecutar 1 ticket	{clientCode, gameCode, sessionId, mode, bet}	{playId, grid0, cascades[], totalWin}
POST	/v1/pack-play	Ejecutar N tickets (para lista + replay)	{clientCode, gameCode, sessionId, mode, bet, packSize}	{packId, plays: [{playId, ticketIndex, grid0, cascades[], totalWin}], totalBet, totalWin, bestIndex}

Definición sugerida de cascades[] (resumen):

- Cada cascada puede incluir: removeCells (lista de posiciones), dropIn (símbolos nuevos por columna) y winStep.
- Opcional: gridAfter para debugging (solo entornos internos).

11. Persistencia y base de datos (consideración)

La persistencia pertenece al backend. Se contempla que el backend definitivo use PostgreSQL o SQL Server (decisión pendiente). El frontend debe ser agnóstico: solo consume APIs.

- Entidades típicas backend: Client/Company, GameConfigVersion, Session, Play (ticket), Pack (paquete), AuditLog.
- El stub puede persistir mínimo o incluso ser stateless para pruebas, según necesidad.

12. Telemetría y diagnóstico (frontend)

- Eventos sugeridos: game_loaded, config_loaded, play_started, play_finished, pack_started, pack_finished, replay_opened, replay_closed, error.
- Incluir correlationId / sessionId en logs para rastreo en backend.
- Métricas: latencia de /play y /pack-play, tasa de errores, FPS promedio (opcional).

13. Criterios de aceptación del MVP

- Los tres modos funcionan end-to-end con backend stub.
- Las cascadas se reproducen fielmente a lo entregado por el backend (sin inventar resultados en cliente).
- Multi-cliente: al cambiar clientCode, cambian parámetros/branding sin modificar código.

- Paquetes: se ejecutan N tickets, se puede abrir replay de cualquier ticket, volver a la lista y elegir otro.
- Resumen del paquete coherente (totalBet/totalWin) y bestIndex consistente con la lista.

14. Consideraciones futuras (no MVP)

- Migración a backend definitivo por microservicios (orquestación, auditoría, wallet, cumplimiento).
- Incorporación de bonus/mini-juegos solo si el roadmap lo habilita (hoy no aplica).
- Herramientas de simulación RTP/volatilidad y certificación.

Fin del documento.