

MÓDULO DE PROYECTO



Técnico Superior en Desarrollo de Aplicaciones Web

Avdaw Games
(Avellaneda Daw Games)

Autor: Jose Javier Ruiz López

Profesor tutor: Pablo Martín Ruiz

Alcalá de Henares, Junio de 2025

índice

1. Introducción y Justificación.....	3
2. Estudio de la viabilidad del proyecto.....	3
2.1. Viabilidad económica.....	3
Retorno de la Inversión (ROI).....	4
Plazo para recuperar la "inversión":.....	4
Escenarios de Tráfico Masivo y Escalado.....	5
Estimación de Costos al Escalar.....	5
1. Vercel (Frontend y Serverless Functions).....	5
2. Supabase (Base de Datos, Autenticación, Almacenamiento, Edge Functions para el backend):.....	6
3. Dominio:.....	8
4. Herramientas y Licencias:.....	8
5. Consideraciones adicionales para el escalado:.....	8
Resumen de Costos Estimados en Caso de Escalado:.....	8
2.2 Viabilidad legal.....	9
Cumplimiento Normativo.....	9
Licencias de Software.....	10
Propiedad Intelectual.....	10
2.3. Viabilidad de Tiempo o Cronograma.....	10
Organización y Planificación Inicial.....	10
Metodología.....	11
Evaluación de las Desviaciones.....	12
3. Análisis y diseño del proyecto.....	13
3.1. Descripción de la Arquitectura Web.....	13
3.2. Tecnologías y Herramientas Utilizadas.....	14
3.3. Análisis de Usuarios (Perfiles de Usuario).....	15
3.4. Definición de Requisitos Funcionales y No Funcionales.....	16
Requisitos Funcionales.....	16
Requisitos No Funcionales.....	17
3.5. Estructura de Navegación (Mapa del Sitio).....	18
3.6. Organización de la Lógica de Negocio.....	19
Estructura de la Lógica Backend (Supabase Services).....	19
Conexión con APIs de Terceros o Servicios Externos.....	20
3.7. Modelo de Datos Simplificado.....	21
Relaciones y estructura de la base de datos.....	23
4. Conclusiones.....	24
Resultados Obtenidos y Cumplimiento de Objetivos.....	24
Retos Encontrados y Soluciones Implementadas.....	24
Aprendizajes y Mejoras Futuras.....	25
Conocimiento en Desarrollo Full-Stack Serverless:.....	25
Aprendizajes Clave y Mejoras Futuras.....	25
5. Bibliografía y fuentes de información.....	26
6. Anexo.....	27

1. Introducción y Justificación

Avdaw Games es una página web de minijuegos cuyo principal objetivo es investigar cómo establecer un **entorno multiusuario sincronizado**, en este caso partidas online y chat en vivo, todo ello sin depender de un servidor dedicado como Node.js.

La motivación detrás de este proyecto surge del deseo de explorar y aprender más allá de lo visto en clase. Me interesaba especialmente probar tecnologías que nunca antes había utilizado, como **Next.js** y **Supabase**. Estas herramientas encajaban perfectamente con la idea de desarrollar un sistema multijugador sin recurrir a un backend monolítico tradicional con React, Node.js y Socket.io, ofreciéndome la oportunidad de aprender algo completamente nuevo mientras construía una aplicación funcional.

2. Estudio de la viabilidad del proyecto

2.1. Viabilidad económica

Dado que el proyecto está construido con Next.js, Supabase, desplegado en Vercel y versionado en GitHub, los costos iniciales de despliegue y mantenimiento actualmente son muy económicos:

Hosting (Vercel): un proyecto como Avdaw Games, actualmente se puede mantener dentro del **plan Hobby de Vercel**, ya que es totalmente **gratuito**.

Este plan ofrece despliegues ilimitados, CDN global, certificados SSL automáticos y un rendimiento excelente para la mayoría de los proyectos personales y pequeños. Vercel también ofrece planes de pago en caso de que el tráfico del sitio se volviese masivo o en caso de necesitar características empresariales.

Base de datos (Supabase): Similar a Vercel, Supabase, ofrece un **plan gratuito** que pese a sus limitaciones, con algún ajuste que se comentara posteriormente, es más que suficiente para empezar.

Este plan incluye una cantidad considerable de base de datos, almacenamiento de archivos, autenticación y funciones *edge* sin costo.

Los gastos aparecerían si tu base de datos creciera exponencialmente o si requirieras características avanzadas de escalabilidad y soporte.

Dominio: Actualmente Vercel te presta un subdominio (vercel.app) gratuito a la hora de desplegar. A largo plazo este es uno de los pocos costos fijos inevitables si se necesita un dominio personalizado (por ejemplo, avdawgames.com). Un dominio .com suele costar entre **10 y 15 € al año**.

Herramientas y Licencias

- **Next.js, Supabase, GitHub:** Todas estas herramientas son **gratuitas y de código abierto** en sus versiones básicas y para la mayoría de los casos de uso.
- **Entorno de desarrollo:** El editor de código, terminal y otras herramientas de desarrollo utilizadas son **gratuitas**.

En resumen, el costo mensual inicial para Avdaw Games actualmente es de 0 € utilizando el subdominio de Vercel, o alrededor de 1-2 € al mes comprando un dominio personalizado.

Retorno de la Inversión (ROI)

Dado que Avdaw Games es una página de minijuegos con un enfoque en la investigación de entornos multiusuario sincronizados y no utiliza un modelo de negocio explícito, el "retorno de la inversión" no se mide en términos económicos directos a corto plazo.

El ROI en este proyecto es principalmente de **aprendizaje y desarrollo de habilidades**:

- **Dominio de nuevas tecnologías:** Aprendizaje de Next.js y Supabase, habilidades muy útiles en el mercado laboral. Pudiendo traducirse en mejores oportunidades laborales o la capacidad de construir proyectos más complejos en el futuro.
- **Experiencia práctica:** La experiencia de construir y desplegar una aplicación completa, gestionar un repositorio de GitHub y entender el ciclo de vida de desarrollo.
- **Portafolio:** Avdaw Games sirve como una demostración de capacidad para desarrollar aplicaciones web modernas y resolver desafíos técnicos.
- **Potencial futuro:** Si bien no hay un plan de monetización actual, podría considerarse como opciones de futuro:
 - **Publicidad:** Integrar anuncios.
 - **Donaciones:** Permitir que los usuarios apoyen el proyecto.
 - **Modelo *freemium*:** Ofrecer juegos básicos gratuitos y algunas características o juegos *premium* de pago.
 - **Venta del proyecto:** En caso de despertar interés para alguien más.

Plazo para recuperar la "inversión"

- El valor del conocimiento adquirido y la mejora del perfil profesional justifican el tiempo y el esfuerzo invertidos, incluso si el retorno monetario directo nunca se materializa.
- Si hablamos de un retorno económico, este sería a **medio o largo plazo**, y dependería de las decisiones que tomes sobre la monetización del sitio.

En caso de querer llevarla la aplicación un paso mas allá la estimación de costes es la siguiente:

Escenarios de Tráfico Masivo y Escalado

Hablar de tráfico masivo para una aplicación como Avdaw Games significa:

- **Altas solicitudes HTTP:** Muchos usuarios accediendo a la web y a los minijuegos.
- **Gran transferencia de datos (bandwidth):** La cantidad de datos (imágenes, código, etc.) que se envía a los usuarios.
- **Intensidad de la base de datos:** Muchas lecturas y escrituras simultáneas debido a las partidas en tiempo real y la gestión de usuarios.
- **Uso de funciones serverless:** Las "partidas online" utilizan funciones *serverless* (Edge Functions en Supabase o Serverless Functions en Vercel) para la lógica del juego o la sincronización, y estas también tienen límites.

Estimación de Costos al Escalar

Pasar de los planes gratuitos a los de pago implica una base mensual y luego costos adicionales basados en el uso.

1. Vercel (Frontend y Serverless Functions)

The screenshot displays three pricing tiers for Vercel. The 'Hobby' plan is free and includes features like CI/CD, WAF, and CDN. The 'Pro' plan costs \$20/month and adds observability tools and faster builds. The 'Enterprise' plan is for critical security and performance, including guest access controls and SLA.

Plan	Description	Price	Key Features
Hobby	The perfect starting place for your web app or personal project.	Free forever.	Import your repo, deploy in seconds; Automatic CI/CD; Web Application Firewall; Global, automated CDN; Fluid compute; DDoS Mitigation; Traffic & performance insights.
Pro	Everything you need to build and scale your app.	\$20/month + additional usage.	Everything in Hobby, plus: 10x more included usage; Observability tools; Faster builds; Cold start prevention; Advanced WAF Protection; Email support.
Enterprise	Critical security, performance, observability, platform SLAs, and support.		Everything in Pro, plus: Guest & Team access controls; SCIM & Directory Sync; Managed WAF Rulesets; Multi-region compute & failover; 99.99% SLA; Advanced Support.

Fuente: <https://vercel.com/pricing>

- **Plan Hobby (Gratis):** Límites generosos para proyectos personales, pero se quedan cortos con tráfico masivo. Por ejemplo, tienen un límite en la duración y memoria de las funciones serverless, y aunque la concurrencia es alta, un uso muy elevado de ancho de banda o de invocaciones a funciones podría exceder los límites.
- **Plan Pro (Recomendado para inicio de escalado):**

- **Costo base:** Unos \$20 (17,51€) al mes por miembro del equipo, en este caso al ser un proyecto individual sería \$20.
- **Plan Enterprise:** Para proyectos gigantes con necesidades específicas y soporte dedicado, esto ya sería para millones de usuarios y un modelo de negocio muy consolidado. Los costos se negocian directamente.

2. Supabase (Base de Datos, Autenticación, Almacenamiento, Edge Functions para el backend)

The image shows the Supabase pricing page with four main plans:












- FREE:** Perfect for passion projects & simple websites. Start for Free. \$0 / month. Includes: Unlimited API requests, 50,000 monthly active users, 500 MB database size (Shared CPU + 800 MB RAM), 5 GB bandwidth, 1 GB file storage, and Community support. Free projects are paused after 1 week of inactivity. Limit of 2 active projects.
- PRO (Most Popular):** For production applications with the power to scale. Upgrade now. From \$25 / month. \$10 in compute credits included. Includes everything in the Free Plan, plus: 100,000 monthly active users (then \$0.00325 per MAU), 8 GB disk size per project (then \$0.125 per GB), 250 GB bandwidth (then \$0.09 per GB), 100 GB file storage (then \$0.021 per GB), Email support, and Daily backups stored for 7 days.
- TEAM:** Add features such as SSO, control over backups, and industry certifications. Upgrade now. From \$599 / month. \$10 in compute credits included. Includes everything in the Pro Plan, plus: SOC2, Project-scoped and read-only access, HIPAA available as paid add-on, SSO for Supabase Dashboard, Priority email support & SLAs, Daily backups stored for 14 days, and 28-day log retention.
- ENTERPRISE:** For large-scale applications running Internet scale workloads. Contact Us. Custom pricing. Includes: Designated Support manager, Uptime SLAs, BYO Cloud supported, 24x7x365 premium enterprise support, Private Slack channel, and Custom Security Questionnaires.

Fuente: <https://supabase.com/pricing>

- **Plan Gratuito**
 - **Base de Datos:** Dispones de **500MB** de almacenamiento para tu base de datos.
 - **Almacenamiento de Archivos:** Tienes **1GB** de almacenamiento disponible para tus archivos.
 - **Ancho de Banda:** Cuentas con **5GB** de ancho de banda.
 - **Usuarios Activos Mensuales (MAUs):** El plan soporta hasta **50.000** usuarios activos mensuales.
 - **Invocaciones de Edge Functions:** Puedes realizar hasta **500.000** invocaciones de Edge Functions.
 - **Consecuencia del Límite de la Base de Datos:** Si el uso de tu base de datos supera el límite de **500MB**, tu proyecto entrará automáticamente en **modo de solo lectura**. Esto significa que no podrás escribir nuevos datos ni realizar cambios en los existentes hasta que actualices tu plan o reduzcas el tamaño de tu base de datos.

- **Consideraciones sobre el Tráfico:** Con un tráfico masivo, es fácil superar estos límites, especialmente los de la base de datos y el ancho de banda, por lo que el proyecto entraría en modo solo lectura.
- **Plan Pro (Recomendado para inicio de escalado)**
 - **Costo base:** \$25(22€) al mes por proyecto.
 - **Base de datos:** 8GB de base de datos incluidos, luego \$0.125(0,11€) por GB adicional.
 - **Almacenamiento de archivos (Storage):** 100GB, luego \$0.021(0,18€) por GB adicional.
 - **Ancho de banda (Bandwidth):** 250GB, luego \$0.09(0,08€) por GB adicional.
 - **Usuarios Activos Mensuales (MAUs):** 100,000 incluidos, luego \$0.00325(0,0028€) por MAU adicional.
 - **Edge Function invocations:** 2 Millones de invocaciones, luego \$2(1,75€) por 1 Million adicional.
 - **Backups diarios y mayor retención de logs.**
- **Cómputo (Compute):** Supabase cobra por la capacidad de cómputo de tu base de datos (CPU/RAM). El plan Pro incluye un crédito de \$10(8,76€) en cómputo. Si la base de datos requiere más recursos debido a muchas consultas complejas o conexiones simultáneas, se puede escalar el cómputo con costos adicionales dependiendo del plan, por ejemplo, \$10(8,76€) al mes para Micro, \$15(13,13€) al mes para Small y \$60(52,53€) al mes para Medium.

A continuación se puede ver la lista completa:

Compute Size	Price USD	CPU	Dedicated	Memory	Connections: Direct	Connections: Pooler
First instance is free on paid plans						
Micro	\$10	2-core ARM		1GB	60	200
Small	\$15	2-core ARM		2 GB	90	400
Medium	\$60	2-core ARM		4 GB	120	600
Large	\$110	2-core ARM		8 GB	160	800
XL	\$210	4-core ARM		16 GB	240	1,000
2XL	\$410	8-core ARM		32 GB	380	1,500
4XL	\$960	16-core ARM		64 GB	480	3,000
8XL	\$1,670	32-core ARM		128 GB	490	6,000
12XL	\$2,800	48-core ARM		192 GB	500	9,000
16XL	\$3,730	64-core ARM		256 GB	500	12,000
>16XL	Contact Us	Custom		Custom	Custom	Custom

- **Costos adicionales por uso:** Similar a Vercel, Supabase tiene un modelo de "pago por uso" si se excede los límites del plan Pro. Por defecto bien eactivado un "spend cap" o límite de gasto, esto puede desactivarse para permitir un escalado ilimitado y mayor control en los gastos

- **Plan Team y Enterprise:** Para proyectos muy grandes, con soporte dedicado, seguridad avanzada y SLA's (Acuerdos de Nivel de Servicio) garantizados. El precio comienza en \$599(524,43€) al mes.

3. Dominio

El costo del dominio personalizado es de unos 10-15€ al año se mantiene, independientemente del tráfico.

4. Herramientas y Licencias

Las herramientas de desarrollo y los frameworks (Next.js, Supabase, GitHub) pese al crecimiento de tráfico siguen siendo gratuitos.

5. Consideraciones adicionales para el escalado

- **Optimización del código:** Es importante la optimización de las consultas a la base de datos, el código de las funciones y el rendimiento general a medida que la app crece para minimizar los costos por uso.
- **Caché:** Implementar estrategias de caché puede reducir significativamente las invocaciones a la base de datos y el uso de ancho de banda.
- **Monitoreo:** Activar las herramientas de monitoreo y *observability* (que Vercel Pro y Supabase Pro ofrecen) son útiles para ayudar a entender dónde se gasta más y a identificar cuellos de botella.
- **CDN (Content Delivery Network):** Vercel ya integra una CDN potente. Para los assets de Supabase Storage, también se benefician de la CDN de Supabase. En este caso no es necesario ontratar una CDN externa adicional a menos que existiesen necesidades muy específicas.

Resumen de Costos Estimados en Caso de Escalado

- **Vercel Pro:** \$20(17,51€) al mes más posibles extras por uso de ancho de banda o funciones.
- **Supabase Pro:** \$25(21,89€) al mes más posibles extras por base de datos, almacenamiento, MAUs, invocaciones de funciones y cómputo de la base de datos.
- **Dominio:** 1.25 € al mes.

Total estimado con tráfico considerable: Nos estaríamos moviendo en un rango de **\$50(43,78€) - \$200(175€) al mes**, dependiendo de qué tan masivo sea ese tráfico y qué tan optimizada esté la aplicación. Los costos pueden aumentar linealmente con el uso, pero las plataformas como Vercel y Supabase están diseñadas para ser rentables incluso a gran escala, ya que solo se paga por lo que se consume.

En conclusión: El modelo de "serverless" y "pago por uso" de Vercel y Supabase es extremadamente eficiente para escalar. Comienzas gratis y, a medida que la

aplicación crece y atrae más usuarios (aumentando las oportunidades de monetización), los costos aumentan, pero de forma controlada y proporcional al valor que generas. Por todo esto, es muy importante monitorizar el uso y entender cómo se distribuyen los costos a medida que el tráfico aumenta.

2.2 Viabilidad legal

La viabilidad legal de Avdaw Games, como cualquier proyecto web que interactúa con usuarios, se centra principalmente en el cumplimiento normativo y la gestión de licencias de software y propiedad intelectual.

Los siguientes apartados pueden consultarse pinchando en los enlaces situados en el pie del sitio web.

© 2025 AWDAW Games. Todos los derechos reservados.

[Terminos del servicio](#) [Política de privacidad](#)

Cumplimiento Normativo

Dado que Avdaw Games es una página web con usuarios, debe cumplir con varias normativas, especialmente si los usuarios provienen de la Unión Europea, donde la regulación de protección de datos es estricta:

- **Protección de Datos Personales (RGPD/GDPR):** Esta es la normativa más importante. Al permitir a los usuarios registrarse, jugar, o si se recopila cualquier tipo de dato que identifique a una persona, se debe cumplir con el Reglamento General de Protección de Datos (GDPR) de la UE. Esto implica:
 - **Política de Privacidad:** Una política clara y accesible que explique qué datos se recopilan, por qué, cómo se usan, cuánto tiempo se guardan y los derechos de los usuarios sobre sus datos.
 - **Consentimiento:** Obtener el consentimiento explícito de los usuarios antes de recopilar sus datos personales.
 - **Seguridad:** Implementar medidas de seguridad adecuadas para proteger los datos de los usuarios.
 - **Derechos de los Usuarios:** Facilitar el ejercicio de derechos como el acceso, rectificación, supresión y portabilidad de sus datos.
- **Política de Cookies:** Como la web utiliza cookies necesita una política de cookies. Esta debe informar a los usuarios sobre los tipos de cookies usadas, su finalidad, y obtener su consentimiento antes de instalarlas (excepto las estrictamente necesarias para el funcionamiento de la web).
- **Comercio Electrónico / LSSI-CE:** Si en el futuro se implementan micropagos, donaciones o cualquier tipo de transacción económica, la Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico (LSSI-CE) española sería aplicable. Esto implicaría, por ejemplo, información fiscal, términos y condiciones de venta, etc.

Licencias de Software

Avdaw Games se beneficia de un ecosistema de desarrollo de código abierto con licencias permisivas:

- **Next.js:** Está bajo la **Licencia MIT**. Esto significa que puede usarse, modificarse, distribuirlo y usarlo comercialmente sin restricciones importantes, siempre que incluyas la notificación de copyright y la licencia.
- **Supabase:** Los componentes de Supabase están bajo diferentes licencias, principalmente la **Licencia MIT** y la **Licencia Apache 2.0**. Ambas son licencias de software libre permisivas que permiten un uso muy amplio, incluso comercial, con atribución.
- **React:** Es la base de Next.js y está bajo la **Licencia MIT**.
- **Dependencias de npm/librerías:** La mayoría de las librerías que usadas en un proyecto Next.js y Supabase están bajo licencias permisivas como MIT, Apache 2.0 o BSD.
- **Imágenes y Recursos Gráficos:** Cualquier imagen, sprite, sonido o recurso gráfico utilizado debe tener una licencia que permita su uso en el proyecto (por ejemplo, Creative Commons con atribución) o ser creaciones propias.

Propiedad Intelectual

Avdaw Games es un proyecto **totalmente original** en cuanto a su implementación y el código fuente desarrollado. No parte directamente de una obra ya existente en su totalidad.

2.3. Viabilidad de Tiempo o Cronograma

Organización y Planificación Inicial

Para el desarrollo de este proyecto, mi organización se centró en una **planificación iterativa y adaptable**, priorizando la flexibilidad ante la complejidad de crear una plataforma de minijuegos multijugador. Aunque no se siguió un cronograma rígido por semanas, el trabajo se estructuró en etapas claras, lo que permitió un avance progresivo y controlado.

Las fases principales y el enfoque temporal para cada una fueron las siguientes:

- **Análisis y Concepción Inicial:** Aquí se definió la visión general del proyecto y las funcionalidades principales. Se exploraron ideas para el proyecto, optando por una plataforma de minijuegos multijugador en lugar de una tienda online como las vistas en clase.
- **Diseño Inicial y Prototipado:** Se procedió al diseño de la página de inicio y se prototipó un primer juego, "Tic Tac Toe", para establecer una estructura básica funcional y probar la integración inicial de elementos.

- **Desarrollo de Juegos Individuales y Diseño de Interfaz Final:** Esta fase se dividió en dos partes. Primero, se desarrolló "Words", un juego para un solo jugador, con el fin de consolidar la lógica de los juegos antes de abordar la complejidad del multijugador. Posteriormente, con dos juegos funcionales, se realizó el diseño estético de las portadas y la interfaz final de la plataforma.
- **Sistema de Autenticación:** Se implementó el sistema de login y registro utilizando **Supabase Auth**. Esta etapa incluyó la adición posterior de **cookies de sesión y validaciones con Zod** para asegurar la robustez y seguridad del acceso de usuarios.
- **Funcionalidades Sociales y Multijugador:** Esta fue la fase más crítica y de mayor dedicación. Se desarrolló una versión inicial del "Tic Tac Toe" multijugador, incorporando **salas de juego y un sistema de chat para poder compartir códigos de sala**. La complejidad de la sincronización en tiempo real hizo que esta fase requiriera un **pulido intensivo** para asegurar una experiencia fluida y estable.
- **Perfiles y Estadísticas de Usuario:** Una vez que la funcionalidad multijugador estuvo estable, se implementaron las secciones dedicadas a los **perfiles de usuario y las estadísticas** de juego, permitiendo a los usuarios visualizar sus estadísticas, puesto y gestionar amistades.
- **Refinamiento y Mejoras Visuales:** La fase final se centró en la corrección de errores menores, el pulido de la interfaz de usuario y la implementación de mejoras visuales generales en toda la plataforma.

Metodología

Aun siendo un trabajo individual, la organización del proyecto se basó en una **metodología ágil y adaptativa**, inspirada en los principios de **Kanban**. No se siguió formalmente **Scrum**, dado que no había un equipo ni "sprints" predefinidos, pero sí se adoptó un enfoque similar de **planificación iterativa basada en funcionalidades**.

- ◆ **Kanban:** metodología ágil que **visualiza el trabajo en un tablero** (generalmente con columnas "Por hacer", "En progreso" y "Hecho") y **limita la cantidad de tareas simultáneas** para optimizar el flujo de trabajo y la productividad.
- ◆ **Scrum:** A diferencia de Kanban, es un **marco de trabajo más prescriptivo** con roles, ciclos de trabajo, eventos y artefactos definidos. Normalmente par equipos

Mi proceso se caracterizó por:

- **Gestión de Tareas:** Mantenía una **lista de tareas** en un cuaderno físico, donde las funcionalidades se clasificaban mentalmente como "Por hacer", "En progreso" y "Hecho".
- **Enfoque en la Finalización:** Me concentraba en completar una funcionalidad o un objetivo específico antes de pasar al siguiente. Este enfoque permitió un progreso tangible y una capacidad de adaptación constante ante los desafíos que surgían.

- **Objetivos Claros:** En lugar de atarme a plazos semanales estrictos, me fijaba objetivos claros y trabajaba para alcanzarlos. Esta flexibilidad fue crucial, especialmente en las fases de mayor complejidad, como la implementación del multijugador.

Evaluación de las Desviaciones

Como es habitual en proyectos de aprendizaje, hubo varias desviaciones respecto a la planificación inicial. Las desviaciones se gestionaron priorizando la consecución del objetivo principal (entorno multiusuario sincronizado sin servidor dedicado) y aprendiendo de los desafíos tecnológicos, lo que llevó a un cronograma más flexible y realista a medida que el proyecto avanzaba.

- **Fases o Tareas que Tardaron Más de lo Esperado:**
 - **Sincronización Multi-usuario:** Esta fue, con diferencia, la tarea que más tiempo consumió. La complejidad de gestionar el estado del juego entre múltiples clientes sin un servidor de juego dedicado implicó mucha investigación, prueba y error, y re-factorización.

Todo esto ocasionó problemas como:

- No poder jugar tu turno.
- Sincronización de los turnos, por ej. al recargar la página se reiniciaba si un jugador es X o O, pudiendo manipular la partida.
- La carga de datos inicial de cada jugador, provocando cosas como la falta de avatar, nombre de usuario y problemas del estilo.
- El almacenamiento de los resultados y las salas a tiempo real

- **Funcionalidades no Completadas o Descartadas:**
 - **Introducción de nuevos juegos online:** Pese a que la aplicación esta preparada para crear salas y manejar diferentes juegos, debido a los problemas anteriores y respetar los tiempos de entrega, se ha optado por no incluir minijuegos que estaban previsto, asegurando así una aplicación mas pequeña pero muy depurada.
 - **Chat en partida:** Inicialmente, pensé en un chat en tiempo real dentro de las partidas. Al ya existir un chat global donde los jugadores pueden hablar, se descartó por priorizar la funcionalidad de juego principal.
 - **Mensajes privados y notificación a tiempo real entre amigos:** el sistema de amigos está implementado, actualmente sirve para seguir los resultados de tus amigos y comparar sus estadísticas con las tuyas desde tu propio perfil. Queda pendiente la capacidad de susurro desde el mismo y avisos a tiempo real para facilitar compartir códigos e información de una partida.

- **Cambio de tema y traducción:** El cambio de tema e idioma requiere añadir un botón en el Header('use client') y pasarle la información al padre, que es el Layout('use server'), gestionarlo en el mismo y devolvérselo a los diferentes Children('use server') para que llamen al diccionario o estilos correspondientes. El problema es, que por razones de seguridad, Next.js no permite que los componentes de uso exclusivo servidor('use server') gestionen información en el navegador(componentes 'use client') obligándote a implementar una serie de requisitos y pasos que implican una gran cantidad de trabajo y refactorización una vez la app está muy avanzada, algo que con otro tipo de Framework sería mucho más sencillo.

3. Análisis y diseño del proyecto

3.1. Descripción de la Arquitectura Web

Avdaw Games se estructura como una **aplicación web moderna y desacoplada**, diseñada para ser escalable y eficiente, especialmente en la gestión de interacciones multiusuario en tiempo real. A nivel general, se compone de dos partes principales que se comunican de forma asíncrona:

- **Frontend (Cliente):** Es la interfaz de usuario, lo que los jugadores ven y con lo que interactúan en su navegador web. Está desarrollado con Next.js y es el responsable de renderizar la página, gestionar las interacciones del usuario y comunicar los eventos del juego al backend.
- **Backend (Servicios Serverless de Supabase):** Aunque no hay un servidor monolítico dedicado tradicional, el backend se materializa a través de los diversos servicios de Supabase. Estos servicios gestionan la base de datos, la autenticación de usuarios, la lógica de negocio (a través de Edge Functions/funciones *serverless*) y, crucialmente, la **sincronización en tiempo real** de las partidas.

Comunicación entre ellas: El **frontend** se comunica con el **backend de Supabase** principalmente a través de:

- **APIs RESTful:** Para operaciones estándar como autenticación, lectura/escritura de datos de usuario o configuración del juego.
- **WebSockets (Supabase Realtime):** Esta es la clave para la experiencia multiusuario. Los jugadores se suscriben a canales en tiempo real, lo que permite que los cambios en el estado del juego se transmitan instantáneamente entre todos los jugadores de una partida, sin necesidad de un servidor intermedio que coordine los mensajes.
- **Edge Functions:** Para ejecutar lógica de negocio específica en el "servidor", como validaciones complejas o procesamiento de datos que no deben ejecutarse en el cliente.

3.2. Tecnologías y Herramientas Utilizadas

La elección de tecnologías para Avdaw Games se centró en la modernidad, la eficiencia y la capacidad de construir un entorno multiusuario sin la complejidad de un servidor backend tradicional.

Frontend:

- **Next.js:** Framework de React para el desarrollo de la interfaz de usuario. Permite *server-side rendering* (SSR) y *static site generation* (SSG), mejorando el rendimiento y el SEO.
- **React:** Librería principal para construir componentes de interfaz de usuario interactivos y reutilizables.
- **Tailwind CSS:** Framework CSS utilitario para un diseño rápido y responsivo, tiene integración con Next.js.
- **JavaScript/TypeScript:** El lenguaje de programación principal, con TypeScript para una mayor robustez y detección de errores en tiempo de desarrollo.

Backend:

- **Supabase:** Es la "columna vertebral" del backend, proporcionando una suite de herramientas *open-source* y *serverless*:
 - **PostgreSQL:** Base de datos relacional robusta y escalable.
 - **Supabase Auth:** Gestión de autenticación de usuarios (registro, login)
 - **Supabase Realtime:** Servicio de WebSockets que permite la suscripción a cambios en la base de datos y la transmisión de eventos en tiempo real, fundamental para la sincronización de las partidas.
 - **Supabase Edge Functions:** Funciones *serverless* basadas en Deno, ideales para lógica de negocio ligera y de bajo latencia, ejecutándose cerca del usuario.
 - **PostgREST:** Genera una API RESTful directamente desde tu base de datos PostgreSQL.
 - **Supabase Storage:** Para el almacenamiento de archivos (como imágenes de juegos, avatares de usuarios, si se implementara).
- **Gmail relay SMTP:** Para suplir los límites de envío de emails en Supabase.

Base de Datos:

- **PostgreSQL (gestionado por Supabase):** Una base de datos relacional potente y flexible. Su elección se debe a la familiaridad, fiabilidad y la integración nativa con Supabase, lo que simplifica la gestión y el escalado.

Integración y Pruebas:

- **GitHub:** Control de versiones y colaboración.
- **Vercel Analytics:** Para monitorizar el rendimiento y el tráfico de la aplicación desplegada.
- **Next.js Dev Tools / React Dev Tools:** Herramientas de depuración para el desarrollo frontend.
- **Supabase Dashboard:** Interfaz web gráfica para gestionar la base de datos, usuarios, tablas y funciones de Supabase.

Seguridad:

- **Supabase Row Level Security (RLS):** Control de acceso a nivel de fila en la base de datos, asegurando que los usuarios solo puedan leer o modificar los datos a los que tienen permiso.
- **Supabase Auth:** Maneja la autenticación de manera segura, incluyendo *hashing* de contraseñas y gestión de sesiones.
- **Cookies de sesión:** Se ha ampliado la autenticación básica de Supabase Auth con cookies personalizadas para tener mayor control y seguridad en la gestión de usuarios.
- **HTTPS:** El despliegue en Vercel automáticamente proporciona certificados SSL/TLS para una comunicación segura.
- **Vercel Security Features:** Vercel incluye protecciones a nivel de red y despliegue para mitigar ataques.

Despliegue y Hosting:

- **Vercel:** Plataforma de despliegue y hosting para aplicaciones Next.js. Ofrece despliegues automáticos desde GitHub, CDN global, y escalabilidad.
- **GitHub:** Repositorio de código fuente. Cada *push* a la rama configurada activa un nuevo despliegue en Vercel.

Otras Herramientas:

- **VS Code:** Editor de código principal.
- **npm:** Gestores de paquetes para dependencias de JavaScript.

3.3. Análisis de Usuarios (Perfiles de Usuario)

Avdaw Games está diseñado principalmente para un perfil de usuario:

- **Jugador (Usuario Autenticado):**
 - **Necesidades:**
 - **Jugar minijuegos online:** Es la necesidad principal ya que quieren acceder a una variedad de juegos simples y divertidos.
 - **Experiencia multijugador fluida:** La sincronización en tiempo real de las partidas es crucial para una experiencia de juego agradable.

- **Interacción social:** Tener la capacidad de jugar, chatear y añadir amistades es un motivador clave.
- **Gestión de cuenta:** Registrarse, iniciar sesión y gestionar el perfil de usuario.
- **Acceso desde cualquier dispositivo:** Que la web sea responsiva y jugable en diferentes pantallas.
- **Privilegios:**
 - Acceder a juegos multijugador.
 - Crear o unirse a partidas.
 - Participar en los minijuegos.
 - Ver tanto sus propias estadísticas y puntuación como las de los demás.
 - Gestionar su información de perfil.

No se prevén perfiles de administrador o moderador en la fase inicial del proyecto, ya que el enfoque central es el juego.

3.4. Definición de Requisitos Funcionales y No Funcionales

Requisitos Funcionales

Estos son las funciones clave que Avdaw Games debe ser capaz de realizar:

- **RF1: Autenticación de Usuarios:**
 - RF1.1: Permitir el registro de nuevos usuarios.
 - RF1.2: Permitir el inicio y cierre de sesión de usuarios existentes.
- **RF2: Gestión de Juegos:**
 - RF2.1: Mostrar una lista de minijuegos disponibles.
 - RF2.2: Permitir a los usuarios seleccionar un minijuego para jugar.
 - RF2.3: Permitir la visualización de estadísticas e información tanto propia como de otros usuarios.
- **RF3: Creación y Unión a Partidas:**
 - RF3.1: Permitir a un usuario crear una nueva partida para un minijuego específico.
 - RF3.2: Permitir a otros usuarios unirse a una partida existente a través de un código de invitación.
 - RF3.3: Mostrar el estado de una partida a tiempo real(jugadores conectados, turnos, etc).
- **RF4: Sincronización de Partidas Multi-usuario:**
 - RF4.1: Sincronizar el estado del juego (movimientos de jugadores, eventos, puntuaciones) en tiempo real entre todos los participantes de una partida.
 - RF4.2: Gestionar el inicio y fin de las partidas.

- RF4.3: Manejar la desconexión inesperada de jugadores y su impacto en la partida.
- **RF5: Interfaz de Usuario:**
 - RF5.1: Proporcionar una interfaz intuitiva para la navegación y el juego que se adapte a cualquier dispositivo.
 - RF5.2: Mostrar mensajes de estado y error al usuario.

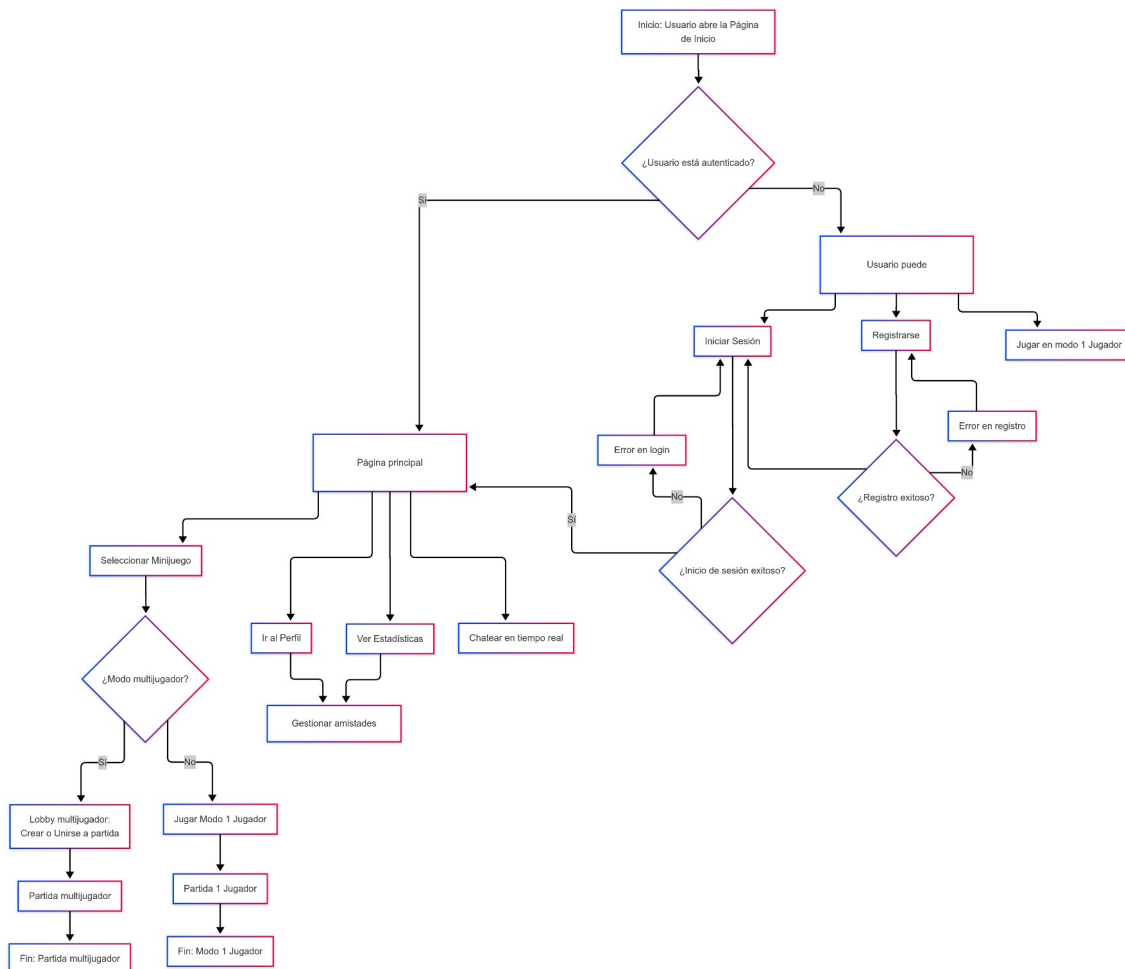
Requisitos No Funcionales

Estos son los aspectos técnicos y de calidad que la aplicación debe cumplir:

- **RNF1: Rendimiento:**
 - RNF1.1: **Latencia Baja:** Las acciones de los jugadores deben reflejarse en tiempo real para todos los participantes con una latencia mínima (idealmente < 100ms para la sincronización crítica).
 - RNF1.2: **Tiempo de Carga:** La página de inicio y los juegos deben cargar rápidamente (idealmente < 3 segundos en conexiones estándar).
 - RNF1.3: **Capacidad de Escalado:** La aplicación debe ser capaz de soportar un número creciente de usuarios concurrentes sin degradación significativa del rendimiento.
- **RNF2: Usabilidad:**
 - RNF2.1: **Intuitividad:** La interfaz de usuario debe ser fácil de entender y usar, incluso para usuarios sin experiencia técnica.
 - RNF2.2: **Claridad:** Los mensajes y la información deben ser claros y concisos.
- **RNF3: Seguridad:**
 - RNF3.1: **Protección de Datos:** Implementar medidas para proteger los datos personales de los usuarios (RGPD)
 - RNF3.2: **Autenticación Segura:** El sistema de autenticación debe ser robusto y proteger contra ataques comunes (ej., fuerza bruta).
 - RNF3.3: **Autorización:** Asegurar que los usuarios solo puedan acceder y modificar los datos a los que tienen permiso.
- **RNF4: Accesibilidad:**
 - RNF4.1: **Diseño Responsivo:** La interfaz debe adaptarse y ser funcional en diferentes tamaños de pantalla (móviles, tabletas, escritorios).
 - RNF4.2: **Contraste de Colores:** Utilizar combinaciones de colores con suficiente contraste para facilitar la lectura.
- **RNF5: Fiabilidad:**
 - RNF5.1: **Disponibilidad:** La aplicación debe estar disponible la mayor parte del tiempo.
 - RNF5.2: **Resiliencia:** La aplicación debe ser capaz de manejar errores y fallos parciales sin colapsar.
- **RNF6: Mantenibilidad:**

- RNF6.1: El código debe ser modular, legible y fácil de mantener y extender.
- RNF6.2: El proyecto debe tener una documentación básica clara.

3.5. Estructura de Navegación (Mapa del Sitio)



Flujo de Navegación Típico:

1. El usuario llega a la **Página de Inicio**.
2. Si no está autenticado, puede **Registrarse**, **Iniciar Sesión** o acceder a juegos de un solo jugador.
3. Una vez autenticado, es redirigido a la página de inicio, donde puede ver la **Lista de Minijuegos** o utilizando la barra de navegación puede acceder a su **Perfil de usuario**, **ver estadísticas** o **chatear a tiempo real con otros jugadores**.
4. En caso de seleccionar un minijuego, multijugador, accede a un “**Lobby**” específico, donde puede crear o unirse a una partida.

3.6. Organización de la Lógica de Negocio

La lógica de negocio de Avdaw Games está distribuida entre el **frontend (Next.js)** y los **servicios de Supabase**, buscando aprovechar al máximo las capacidades de cada uno para una arquitectura *lean* y orientada al tiempo real.

Estructura de la Lógica Backend (Supabase Services)

Gracias al funcionamiento de Next.js podemos tener componentes de tipo servidor(actions.ts) o de tipo cliente(page.tsx) que se comuniquen entre sí, esto permite que sin tener un servidor backend "monolítico" la lógica se reparta entre los servicios de Supabase:

PostgreSQL (Base de Datos):

- **Almacenamiento de Datos:** Contiene el estado de la aplicación (usuarios, partidas, información de juegos).
- **Row Level Security (RLS):** Aquí se define gran parte de la **lógica de autorización**. Por ejemplo, una política de RLS puede asegurar que un usuario solo pueda ver las partidas en las que participa o modificar sus propios datos. Esto es una forma muy potente de externalizar lógica de seguridad.
- **Triggers/Funciones SQL (PostgreSQL Functions):** Para lógica más compleja que debe ejecutarse directamente en la base de datos tras ciertos eventos (ej., actualizar el estado de una partida automáticamente cuando un jugador se une o desconecta).

Supabase Realtime:

- **Sincronización de Juego:**
Es la pieza central para la lógica de negocio multiusuario. Los clientes se suscriben a canales (salas), cuando un jugador realiza una acción (turno), el cliente actualiza el estado de la base de datos (o emite un evento como almacenar en la tabla rooms el estado de la partida). Realtime detecta estos cambios y los propaga instantáneamente a todos los demás clientes suscritos a ese canal.

- **Presencia:** Gestiona qué usuarios están activos en una sala o partida, permitiendo saber quién está conectado.

```
channel
.on(
  REALTIME_LISTEN_TYPES.POSTGRES_CHANGES,
  {
    event: REALTIME_POSTGRES_CHANGES_LISTEN_EVENT.UPDATE,
    schema: 'public',
    table: 'rooms',
    filter: 'room_name=eq.${roomName}',
  },
  handleGameUpdate
)
.on(
  REALTIME_LISTEN_TYPES.PRESENCE,
  { event: 'sync' },
  handlePresenceSync
)
.subscribe(async (status) => {
  if (status === 'SUBSCRIBED') {
    console.log('Suscrito a la sala:${roomName}');
    await channel.track({ name: currentUser });
  }
  if (status === 'CHANNEL_ERROR') {
    console.error('Error en canal de Supabase:', channel.state);
  }
});

return () => {
  console.log('Desuscrito de la sala:${roomName}');
  supabase.removeChannel(channel);
};
}, [roomName, currentUser]);
```

Supabase Edge Functions:

- **Lógica de Negocio Específica:** Para aquellas operaciones que requieren más procesamiento o validación de lo que RLS puede ofrecer, o que no deben ejecutarse en el cliente. Ejemplos:
 - **Creación de Partidas Complejas:** Lógica para asignar IDs de partida, verificar disponibilidad, etc.
 - **Validación de Movimientos:** En juegos más complejos, una función Edge podría validar si un movimiento es legal antes de actualizar la base de datos.
 - **Cálculos de Puntuación Final:** Lógica para calcular la puntuación total de una partida al finalizar.
 - **Webhooks:** Si Avdaw Games necesitara integrarse con servicios externos.

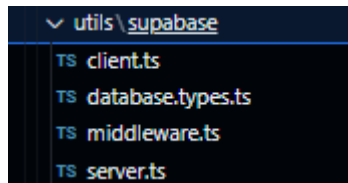
Supabase Auth:

- **Lógica de Autenticación:** Gestiona el registro, login, tokens de sesión y restablecimiento de contraseña. Aunque es un servicio, su configuración implica una lógica de negocio fundamental.

Conexión con APIs de Terceros o Servicios Externos

Actualmente, Avdaw Games se centra en la funcionalidad core, es decir, depende de sí misma, y su interacción principal es con los servicios de **Supabase**. Esto significa que no se conecta con APIs de terceros o servicios externos significativos como pasarelas de pago o APIs públicas de forma directa.

- **Supabase como Servicio Externo Principal:** Se podría considerar a Supabase como el principal "servicio externo" con el que interactúa la aplicación, ya que es una plataforma en la nube que provee la base de datos, autenticación y capacidades en tiempo real. La interacción se realiza a través de su SDK (supabase-js) integrándolo en el frontend y las Edge Functions en el backend.



Si en el futuro se planea la monetización (donaciones, compras), se integrarían:

- **Pasarelas de Pago:** Servicios como Stripe o PayPal para procesar transacciones monetarias. Esto implicaría una lógica adicional en las Edge Functions para gestionar la creación de pagos, la confirmación y la actualización del estado en la base de datos.

3.7. Modelo de Datos Simplificado

Avdaw Games organiza su información en las siguientes tablas principales, utilizando PostgreSQL gestionado por Supabase:

1. users (Usuarios) Representa los perfiles de los jugadores registrados en la aplicación.

- **id:** UUID(Clave Primaria, generada por Supabase Auth)
- **username:** VARCHAR (Nombre de usuario para identificación pública)
- **email:** VARCHAR(Correo electrónico del usuario, para autenticación)
- **avatar:** TEXT(Campo para la URL de un avatar en el Storage de Supabase)
- **descripcion:** TEXT(Descripción o "acerca de mí" del usuario)
- **amigos:** JSONB (Almacena una lista de IDs de usuario que son "amigos", permitiendo relaciones entre usuarios)
- **favorito:** TEXT(Campo para almacenar juego favorito)
- **created_at:** TIMESTAMPTZ(Marca de tiempo de creación del registro del usuario)

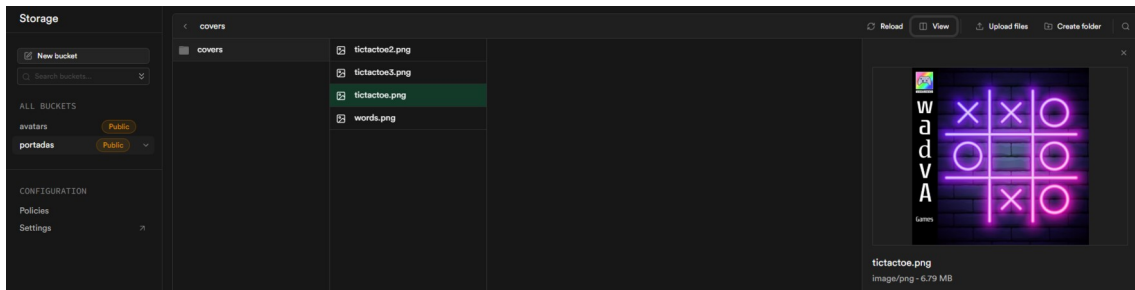
2. games (Juegos/Minijuegos) Define y cataloga los diferentes minijuegos disponibles en la plataforma, sirve para generar las tarjetas de la página principal.

- **id:** UUID(Clave Primaria, generada por Supabase)
- **nombre:** VARCHAR(Nombre legible del minijuego, ej., "Tic-Tac-Toe")
- **descripcion:** TEXT(Explicación breve del juego)
- **game_id:** VARCHAR(Un identificador único de cadena para el juego, sirve para la redirección de los juegos single player)

```
<Link href={`/${juegos/${juego.game_id}}`>
```

- **imagen**: TEXT(Ruta a la imagen representativa del juego en el Storage*)
- **modo**: VARCHAR(Tipo de juego, ej., "2 jugadores", "PvP")
- **pathPrefix**: TEXT(Parecido a game_id, sirve para crear la ruta de redirección de los juegos multijugador)

```
<Link href={`/${juego.pathPrefix}/${juego.game_id}`}>
```



*Muestra del Storage de Supabase.

3. rooms (Salas de Juego/Partidas) Esta tabla es fundamental para la gestión de las partidas en tiempo real. Cada registro representa una instancia de una partida que se está creando o está en curso.

- **id**: UUID(Clave Primaria, generada por Supabase)
- **created_at**: TIMESTAMPTZ (Marca de tiempo de creación de la sala)
- **room_name**: TEXT(Nombre de la sala, necesario para unirse a una partida, generado aleatoriamente.)

```
import {uniqueNamesGenerator, colors} from 'unique-names-generator'

const numberDictionary = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
const randomName: string = uniqueNamesGenerator({
  dictionaries: [colors, numberDictionary, numberDictionary, numberDictionary, numberDictionary, numberDictionary],
  separator: '',
  length: 6,
})
```

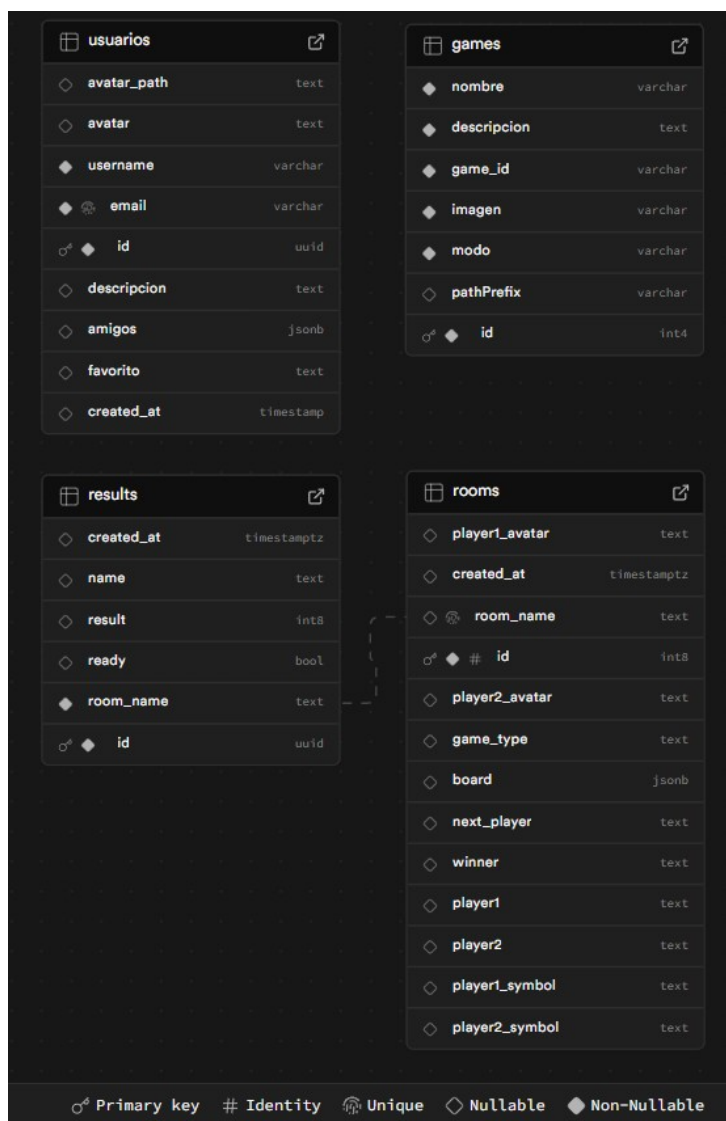
Código: olive24049

- **game_type**: TEXT(Por si en un futuro se añaden más minijuegos, poder identificar a que juego pertenece la sala creada ej. tictactoe)
- **board**: JSONB(Almacena el estado actual del juego en formato JSON. **Este es el dato clave para la sincronización en tiempo real.** Contendrá el tablero, turno actual, etc., específico para cada tipo de minijuego.)
- **next_player**: TEXT(Indica el username del jugador al que le corresponde el siguiente turno)
- **player1_avatar**: TEXT(URL del avatar del jugador 1 en esta partida)
- **player1_symbol**: TEXT (Símbolo o pieza del jugador 1 en el juego, ej., "X")
- **player2_avatar**: TEXT(URL del avatar del jugador 2 en esta partida)
- **player2_symbol**: TEXT(Símbolo del jugador 2 en el juego, ej. "O")
- **winner**: TEXT(Almacena el username del jugador ganador al finalizar la partida, si aplica)

4. results (Resultados de Partida) Registra los resultados finales de las partidas jugadas, sirviendo como historial.

- **id**: UUID(Clave Primaria)
- **created_at**: TIMESTAMPTZ(Marca de tiempo en que se registró el resultado)
- **name**: TEXT(username del usuario para almacenar su resultado en una sala)
- **result**: JSONB(Contiene un número dependiendo de si el usuario ha perdido, empatado, ganado o no existe un resultado, siendo en orden -1,0,1,2)
- **room_name**: TEXT(Para vincular el resultado a la partida específica.)
- **ready**: BOOLEAN(En caso de que un juego necesite confirmación por parte de todos los jugadores para empezar la partida, cambiaría a true y comenzaría la misma).

Relaciones y estructura de la base de datos



4.Conclusiones

Resultados Obtenidos y Cumplimiento de Objetivos

El desarrollo de este proyecto culminó con la creación de una **plataforma funcional de minijuegos multijugador en tiempo real**, lo que representa un éxito significativo en el cumplimiento de los objetivos planteados inicialmente. Se logró implementar un sistema robusto de autenticación de usuarios, una interfaz de usuario atractiva y, lo más importante, la **sincronización multiusuario efectiva** para los juegos.

Específicamente, los objetivos clave alcanzados incluyen:

- **Implementación de un entorno multiusuario sin servidor dedicado:** Se logró utilizar tecnologías serverless (Next.js y Supabase) para gestionar usuarios y la lógica de juego, validando la viabilidad de este enfoque.
- **Desarrollo de un sistema de registro y login robusto:** La autenticación basada en Supabase Auth, complementada con cookies de sesión y validaciones con Zod, proporciona una base segura para los usuarios.
- **Creación de minijuegos funcionales:** Tanto "Tic Tac Toe" como "Words" se desarrollaron con éxito, sentando las bases para la expansión futura de la biblioteca de juegos.
- **Sincronización de partidas multijugador:** El desafío más grande, la implementación del "Tic Tac Toe" multijugador con salas y chat, se resolvió satisfactoriamente utilizando las capacidades de Realtime de Supabase, demostrando la capacidad de la plataforma para gestionar interacciones en tiempo real.
- **Interfaz de usuario intuitiva y atractiva:** El diseño y la estética final de la interfaz contribuyen a una experiencia de usuario positiva y coherente.
- **Perfiles de usuario y estadísticas básicas:** Se integraron funcionalidades sociales que enriquecen la experiencia del usuario al permitirles ver su progreso.

A pesar de las desviaciones en la planificación, el foco se mantuvo en la robusted de las funcionalidades esenciales, resultando en un producto principal estable y operable.

Retos Encontrados y Soluciones Implementadas

Durante el proceso de desarrollo, surgieron varios retos que requirieron soluciones innovadoras y un enfoque flexible:

- **Reto 1: Complejidad de la Sincronización en Tiempo Real.**
 - **Descripción:** Como he mencionado en el apartado 2.3, la mayor dificultad radicó en conseguir una sincronización funcional para la gestión de estados de partida y movimientos de los jugadores.

- **Solución Implementada:** Se utilizó la funcionalidad **Realtime de Supabase** para la comunicación bidireccional y la actualización instantánea de los estados de juego.
- **Reto 2: Gestión de Sesiones y Seguridad de Autenticación.**
 - **Descripción:** Asegurar que el sistema de login y registro fuera no solo funcional, sino también seguro, persistente y adaptable a futuras necesidades a través de las sesiones del usuario.
 - **Solución Implementada:** Además de la autenticación básica de Supabase Auth, se implementaron **cookies de sesión** para mantener el estado del usuario. La inclusión de **validaciones con Zod** en el lado del cliente y del servidor añadió una capa crucial de seguridad y robustez a los formularios de registro y login, previniendo entradas de datos inválidas y posibles vulnerabilidades.

Aprendizajes y Mejoras Futuras

El desarrollo de este proyecto ha proporcionado aprendizajes significativos en diversas áreas:

- **Conocimiento en Desarrollo Full-Stack Serverless:**
El proyecto ha consolidado un **profundo conocimiento en el desarrollo en este tipo de arquitectura**, utilizando **Next.js y Supabase**. Esta experiencia ha demostrado ser una alternativa potente y ágil a las arquitecturas tradicionales, ofreciendo una nueva perspectiva en la creación de aplicaciones web modernas. Se ha adquirido una valiosa competencia al aprender a integrar de manera eficiente y diferente a lo habitual servicios esenciales como **bases de datos, autenticación de usuarios y funcionalidades en tiempo real**.
- **Dominio de la lógica de sincronización en tiempo real:**
La fase del multijugador fue la de mayor aprendizaje técnico, proporcionando una comprensión profunda de los desafíos y soluciones asociados con la sincronización de estados de juego entre múltiples clientes.
- **Aprendizajes Clave y Mejoras Futuras**
El desarrollo de este proyecto ha sido fundamental para **mejorar mi capacidad de planificación y estructuración** en futuros trabajos. Esta experiencia me ha enseñado a anticipar y **evitar complicaciones** que, en esta ocasión, impidieron la implementación de funcionalidades deseables como un gestor de temas o traducciones.

De cara a futuras mejoras y expansiones del proyecto, se plantean las siguientes líneas:

- **Expansión de la Biblioteca de Minijuegos:**
Añadir más juegos, explorando diferentes géneros y complejidades, y posiblemente una estructura que facilite la contribución de la comunidad.

- **Mejora de las Funcionalidades Sociales:**
ampliar el sistema de amigos, susurros e invitaciones, y la integración del chat en la partida.
- **Optimización del Rendimiento y Escalabilidad:**
A medida que la base de usuarios crezca, será crucial revisar y optimizar el rendimiento de la base de datos y la lógica de tiempo real para asegurar una experiencia fluida.
- **Monetización:**
Explorar modelos de monetización sencillos para que sea viable un mantenimiento y expansión.
- **Refinamiento de la Interfaz de Usuario/Experiencia de Usuario:**
Recopilar feedback de usuarios para realizar mejoras continuas en la facilidad de uso y el diseño visual.

5. Bibliografía y fuentes de información

Configuración cliente/ servidor NextJs

<https://nextjs.org/docs/app/getting-started/layouts-and-pages>

Integración Supabase Auth

<https://supabase.com/docs/guides/auth/server-side/nextjs>

Despliegue

[Deploying to Vercel](#)

Componentes de Supabase integrados

- **Chat:** <https://supabase.com/ui/docs/nextjs/realtime-chat>
- **Avatar:** [Current User Avatar](#)
- **Cursores:** [Realtime Cursor](#)

Generador de ids:

<https://www.npmjs.com/package/unique-names-generator>

Iconos

<https://react-icons.github.io/react-icons>

Base para la creación del Perfil de usuario

<https://www.creative-tim.com/twcomponents/component/profile-card-horizon-ui-tailwind>

Servidor SMTP

[Enviar correo electrónico desde impresoras, escáneres o aplicaciones - Ayuda de Administrador de Google Workspace](#)

Crédito y Agradecimientos

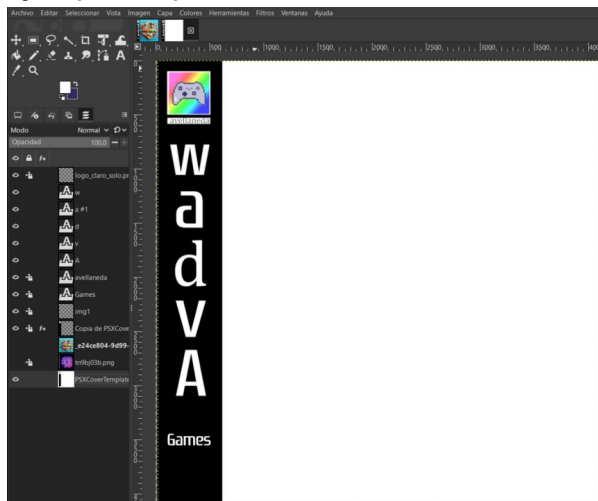
- **Creación de salas Multijugador:**
[Exploring Supabase Realtime By Building a Game](#)
- **Autenticación y cookies personalizadas:**
[Next.js: Authentication \(Best Practices for Server Components, Actions, Middleware\) - YouTube](#)

Recursos gráficos:

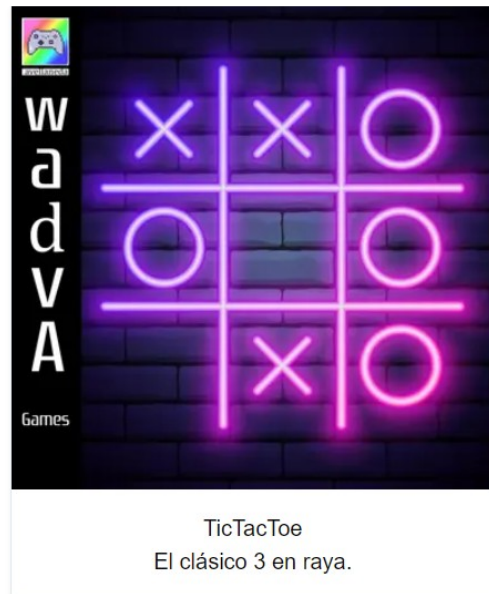
Para el logo y las portadas tenía claro que quería darle un estilo retro, al final esta idea terminó derivando a un estilo homenaje a PlayStation 1.

Basándome en bocetos generados por IA, tanto el logo como las portadas han sido editados y montados a mano utilizando [GIMP - GNU Image Manipulation Program](#), creando una plantilla adaptable a ampliaciones.

Ejemplo de plantilla.



Resultado final.



6. Anexo

La **documentación técnica**, el **manual de usuario** y las **instrucciones de instalación** están disponibles para su consulta en el repositorio de GitHub del proyecto: [jrlpz/avdaw_games](https://github.com/jrlpz/avdaw_games)