Resumen Ejecutivo de la Visión HyPanel

1. ¿Qué es HyPanel?

Imagina **HyPanel** como el "Shopify" o el "Wix" para los servidores de Hytale. Actualmente, crear un servidor de juegos requiere conocimientos técnicos (lidiar con archivos, puertos y configuraciones manuales). Tu visión es democratizar esto: convertir la creación de un mundo virtual en algo tan sencillo como instalar una aplicación en tu teléfono.

2. El Problema Actual (El Dolor del Usuario)

- **Es caro e ineficiente:** Hoy en día, si quieres un servidor, pagas una mensualidad fija (por ejemplo, $10/mes) aunque nadie esté jugando. Estás pagando por aire el 80% del tiempo.
- **Es difícil:** Instalar "mods" (modificaciones) suele requerir mover archivos manualmente y rezar para que no rompan el juego.
- **Está fragmentado:** Tienes que buscar los mods en una web, pagar el servidor en otra y configurar los permisos en una tercera.

3. La Solución "Mágica" de HyPanel

A. Servidores que "Duermen" (Eficiencia Total)

En lugar de alquilar un servidor que está encendido las 24 horas gastando dinero, HyPanel ofrece **servidores bajo demanda**.

- **Cómo se siente:** Cuando tú o tus amigos queréis jugar, hacéis clic en conectar. El servidor "despierta" instantáneamente (en menos de un segundo).
- **El Beneficio:** Permite tener cientos de mundos creados sin arruinarse. Solo cuestan dinero cuando realmente hay gente divirtiéndose dentro.

B. Hytahub: La "App Store" de tu Mundo

Integramos una tienda de contenidos directamente en el panel de control.

- **Experiencia "Plug & Play":** ¿Quieres añadir dragones a tu mundo? Buscas "Dragones" en el panel, le das a "Instalar" y listo.
- **Sin descargas para el jugador:** Gracias a cómo funciona Hytale, tus jugadores no tienen que instalar nada. Al entrar a tu servidor, el juego descarga automáticamente todo lo necesario. Es una experiencia 100% fluida.

C. "La Forja": Crear sin Programar

Hytale ha eliminado la necesidad de escribir código complejo para diseñar aventuras. HyPanel aprovecha esto con un editor visual en la web.

- **Diseño Visual:** Imagina una pizarra blanca donde conectas cajas con líneas. Una caja dice "Al entrar el jugador" y la conectas con una línea a otra que dice "Dar espada de fuego".
- **Poder para todos:** Permite que cualquier persona creativa diseñe minijuegos, misiones o historias complejas desde su navegador, sin ser programador.

4. Modelo de Negocio (Todos Ganan)

HyPanel no es solo una herramienta, es un **ecosistema económico**.

- **Para los Creadores de Mods:** Pueden vender sus creaciones (paquetes de armas, mapas, monstruos) en Hytahub.
- **Para los Dueños de Servidores:** Tienen acceso a contenido profesional fácil de instalar y herramientas para cobrar suscripciones o vender pases de batalla a sus jugadores.
- **Para HyPanel:** Gana dinero por el alojamiento (hosting) y se lleva una pequeña comisión de las transacciones del mercado, retroalimentando el sistema.

5. ¿Por qué ahora?

Hytale está diseñado desde cero para permitir exactamente esto (algo que Minecraft nunca pudo hacer bien). Al posicionarte como la plataforma que conecta **la tecnología** (servidores), **el contenido** (Hytahub) y **la creatividad** (La Forja) en un solo lugar, resuelves todos los dolores de cabeza de la comunidad antes de que el juego siquiera salga al mercado masivo.

En resumen: HyPanel hace que tener un servidor profesional sea tan fácil, barato y divertido como jugar al propio juego.

HyPanel: Plan de Proyecto e Infraestructura Técnica

Versión del Documento: 2.0 (Post-Anuncio "Hytale is Saved")
Objetivo: Definir la arquitectura, modelo de negocio y hoja de ruta para una plataforma de gestión de servidores elástica ("Serverless") para Hytale.

---

1. El Nuevo Panorama Técnico: "Legacy Engine"

Antes de diseñar la solución, debemos establecer los hechos técnicos inamovibles sobre los que se construirá HyPanel. Tras la cancelación del motor C++ y el retorno al motor original, estas son las reglas del juego:

1. **El Servidor es Java:** El servidor dedicado oficial corre sobre la JVM (Java Virtual Machine). Esto significa que tenemos acceso a un ecosistema maduro, pero heredamos la "pesadez" de Java (consumo de RAM y tiempos de arranque lentos).[1]
2. **Protocolo de Red = QUIC (UDP):** Hytale ha abandonado TCP para el tráfico de juego en favor de **QUIC** (un protocolo fiable sobre UDP). Esto es crítico: los proxies tradicionales de Minecraft (como BungeeCord o Velocity estándar) **no funcionarán** porque no entienden QUIC.[3]
3. **Modding "Server-Side First":** El cliente no se toca. Cuando un jugador entra, el servidor le envía los archivos. HyPanel solo necesita gestionar los archivos del servidor; el cliente se actualiza solo.[4]
4. **Lógica Visual:** No habrá scripting de texto (Lua/JS) para diseñadores. La lógica del juego (misiones, IA) se define mediante **Visual Scripting** (nodos) y archivos JSON.[4]

---

2. Arquitectura del Sistema: "La Nube Elástica"

La visión de "servidores que duermen y despiertan instantáneamente" (Scale-to-Zero) choca con la realidad de Java (que tarda 15-30s en arrancar). Para lograr tu visión, HyPanel implementará una arquitectura de tres capas con una tecnología secreta: **CRaC**.

2.1 La Puerta de Entrada: El "Void Proxy" (Capa QUIC)

Como el protocolo es QUIC, necesitamos un proxy de nueva generación. No podemos usar Nginx estándar fácilmente para lógica de juego.

- **Tecnología Recomendada: Rust** (basado en Quilkin de Google/Embark) o **Go** (usando quic-go).[6]
- **Función:**
  1. Recibe la conexión UDP/QUIC del jugador.
  2. Lee el paquete de "Handshake" para ver a qué servidor quiere ir (ej. lobby.hypanel.com).
  3. Consulta al "Cerebro" (Redis) si ese servidor está vivo.
  4. **Si está dormido:** El Proxy **retiene** la conexión. Envía paquetes de "Keep-Alive" al cliente para que no se desconecte, mostrando quizás una pantalla de carga o partículas, mientras ordena al backend despertar.
  5. **Si está despierto:** Enruta los paquetes UDP directamente al contenedor Docker.

2.2 La Magia de la Elasticidad: Java + CRaC

Aquí es donde HyPanel se diferencia de cualquier competencia.

- **El Problema:** Un servidor Java normal tarda demasiado en encender. El jugador se iría.
- **La Solución:** Usar **OpenJDK CRaC (Coordinated Restore at Checkpoint)**.[8]
  - *Cómo funciona:* HyPanel arranca un servidor Hytale "plantilla", carga los plugins básicos y el mapa, y luego "congela" la memoria RAM en un archivo en el disco (Checkpoint).
  - *El Despertar:* Cuando el Proxy pide un servidor, HyPanel no arranca Java desde cero. **Restaura** la memoria desde el disco.
  - *Resultado:* Tiempo de inicio: **< 500 milisegundos**. El jugador ni siquiera nota que el servidor estaba apagado.

2.3 Contenedores y Orquestación

- **Docker:** Cada mundo/servidor es un contenedor aislado.
- **Sidecar ("El Agente"):** Un pequeño programa (escrito en Go) corre junto a cada servidor. Su trabajo es escuchar a **Hytahub** y descargar mods/assets en caliente sin reiniciar el servidor, aprovechando la capacidad de "Hot-Reload" de Hytale.[3]

---

3. Hytahub: El Repositorio Centralizado

Hytahub no es solo una web, es el sistema de archivos remoto de tus servidores.

3.1 Estructura de Datos

Dado que Hytale se basa en datos (JSON), Hytahub funcionará como un registro de dependencias (similar a npm o Maven pero para juegos).

- **Assets:** Modelos .gltf o .bbmodel (Blockbench).
- **Configuraciones:** Archivos .json que definen el comportamiento de mobs, tablas de loot, etc.
- **Plugins:** Archivos .jar (Java) para lógica compleja del servidor.

3.2 Flujo de Monetización (Revenue Share)

El sistema se inspira en **Tebex** pero integrado nativamente.

1. **Creador:** Sube un "Pack de Dragones" a Hytahub. Precio: $10.
2. **Dueño de Servidor:** Compra el pack en el panel.
3. **HyPanel:**
    - Cobra los $10.
    - Asigna la licencia al servidor del comprador.
    - El "Agente Sidecar" descarga los archivos *al servidor* automáticamente.

    - Reparte los ingresos (ej. 70% Creador / 30% HyPanel).
4. **Seguridad:** Como los archivos van de Hytahub -> Servidor Docker (que tú controlas), el usuario final (jugador) nunca toca los archivos fuente, protegiendo la propiedad intelectual del creador.

---

4. "La Forja": Editor Visual Web

Hytale ha confirmado que **no usará scripting de texto** para el diseño, sino nodos visuales. HyPanel debe replicar esto en la web para permitir configurar servidores desde el navegador.

4.1 Tecnología: React Flow

Usaremos **React Flow** para crear un editor de nodos en el navegador.[10]

- **¿Por qué?** Permite crear interfaces tipo "diagrama" donde conectas cajas con cables. Es rápido, moderno y muy personalizable.
- **Funcionalidad:**
    - El usuario abre "La Forja" en HyPanel.
    - Arrastra un nodo "Evento: Al Morir".
    - Lo conecta a un nodo "Acción: Soltar Moneda".
    - Al guardar, La Forja genera el archivo JSON exacto que el servidor de Hytale necesita leer.

---

5. Hoja de Ruta (Roadmap)

Este plan asume que el acceso anticipado de Hytale llegará en 2026, dándonos tiempo para desarrollar la infraestructura.

Fase	Duración	Objetivo Principal	Entregables Clave
Fase 1: I+D Red y Java	Mes 1-3	Prototipo de "Wake-on-LAN"	1. Proxy QUIC (Rust/Go) funcional.

			2. Demo de Java con CRaC arrancando en <1s.
Fase 2: El Core (Panel)	Mes 4-6	Gestión de Contenedores	1. Panel Web básico (React). 2. Orquestador que crea/destruye Dockers. 3. Integración básica de Redis.
Fase 3: Hytahub & Assets	Mes 7-9	Sistema de Archivos	1. Marketplace Hytahub (Backend). 2. Agente Sidecar para inyección de archivos. 3. Sistema de cuentas y pagos.
Fase 4: La Forja	Mes 10-12	Editor Visual	1. Editor de nodos (React Flow) para generar JSONs de Hytale. 2. Integración con Blockbench Web.
Fase 5: Beta Privada	Mes 13+	Pruebas de Carga	Lanzamiento con creadores de contenido selectos para poblar Hytahub antes del juego.

---

6. Resumen de Stack Tecnológico Definitivo

Componente	Tecnología	Razón de la elección
Proxy (Red)	**Rust (Quilkin)** o Go	Soporte nativo de QUIC/UDP de alto rendimiento.
Panel Backend	**Go** o Node.js	Escalabilidad y facilidad para trabajar con Docker.
Frontend	**React** + Tailwind	Estándar de industria, ecosistema rico (React Flow).
Editor Visual	**React Flow**	Librería líder para crear editores de nodos en web.
Base de Datos	**PostgreSQL** (Datos) + **Redis** (Estado)	Fiabilidad + Velocidad en tiempo real.
Optimización Java	**Azul Zulu JDK + CRaC**	Única forma viable de lograr arranque instantáneo en Java.

Conclusión

Tu idea es técnicamente viable y comercialmente potente. El cambio a **Legacy Engine (Java)** y **QUIC** ha creado una barrera de entrada técnica que la mayoría de hostings tradicionales no podrán superar fácilmente (sus paneles están hechos para TCP y servidores siempre encendidos).

Al construir **HyPanel** sobre **CRaC** (para eliminar el coste de RAM en reposo) y un **Proxy QUIC** inteligente, tendrás una ventaja competitiva masiva: podrás ofrecer servidores "gratuitos" o muy baratos que solo consumen recursos cuando se usan, financiados por las transacciones

del mercado **Hytahub**.

Fuentes citadas

1. An overview of Hytale's server technology, acceso: noviembre 27, 2025, https://hytale.com/news/2019/1/an-overview-of-hytales-server-technology
2. Hytale's New Engine Update - YouTube, acceso: noviembre 27, 2025, https://www.youtube.com/watch?v=ulWxubsuinc
3. All the Questions and Answers from the Q&A that took place on the Hytale Discord Server. : r/HytaleInfo - Reddit, acceso: noviembre 27, 2025, https://www.reddit.com/r/HytaleInfo/comments/1p464cu/all_the_questions_and_answers_from_the_qa_that/
4. Hytale Modding Strategy and Status, acceso: noviembre 27, 2025, https://hytale.com/news/2025/11/hytale-modding-strategy-and-status
5. Hytale Modding: Current State and Future Vision – What You Need to Know!, acceso: noviembre 27, 2025, https://hytale.game/en/hytale-modding-current-state-and-future-vision-what-you-need-to-know/
6. Introducing Quilkin: the open-source game server proxy | Google Cloud Blog, acceso: noviembre 27, 2025, https://cloud.google.com/blog/products/gaming/introducing-quilkin
7. A production-ready QUIC implementation in pure Go - GitHub, acceso: noviembre 27, 2025, https://github.com/quic-go/quic-go
8. Checkpoint and Restore With the JVM :: Spring Boot, acceso: noviembre 27, 2025, https://docs.spring.io/spring-boot/reference/packaging/checkpoint-restore.html
9. Fast application startup with OpenJDK CRaC - Foundations - Ubuntu Discourse, acceso: noviembre 27, 2025, https://discourse.ubuntu.com/t/fast-application-startup-with-openjdk-crac/63151
10. React Flow: Node-Based UIs in React, acceso: noviembre 27, 2025, https://reactflow.dev/
11. Tools for building a Graph/Node based user interface in a webapp - Stack Overflow, acceso: noviembre 27, 2025, https://stackoverflow.com/questions/72164885/tools-for-building-a-graph-node-based-user-interface-in-a-webapp