Arquitectura Técnica - Juego Multijugador en Tiempo Real

Propuesta de diseño escalable y distribuido

Hecho por Erick Bonilla.

Problema y Objetivos

- Diseñar un sistema multijugador en línea en tiempo real que garantice:
- - Alta concurrencia y baja latencia (<200ms)
- - Escalabilidad horizontal y tolerancia a fallos
- - Consistencia y fiabilidad en datos críticos
- - Seguridad en autenticación y prevención de trampas

Stack Tecnológico Propuesto

- Frontend: vanilla Javascript Cliente.
- Backend: Game Servers en Rust (server-authoritative)
- API Gateway con TLS Offload
- Autenticación: Keycloak (OAuth2/OpenID, Google/Facebook)
- Almacenamiento: PostgreSQL (transacciones/perfiles) + Redis (estado efímero, matchmaking)
- Infraestructura: K3s
- Observabilidad: Prometheus + Grafana

Mecanismos de Escalabilidad

- - Game Gateway enruta jugadores a múltiples instancias
- - Autoescalado basado en número de jugadores
- - Redis Cluster para manejo de matchmaking y presencia
- - PostgreSQL con replicas para lectura
- - Service discovery (Consul)

Tolerancia a Fallos

- - Snapshots efímeros en Redis con TTL para recuperación
- - Balanceo de carga con soporte WebSockets persistentes
- - Game servers aislados (fallo de uno no afecta a otros)
- - API Gateway con rate limiting (IP/usuario)

Seguridad y Privacidad

- - Autenticación híbrida: Cuentas propias + Google/Facebook
- - Validación de cuentas vía OTP
- - Server-authoritative para evitar trampas
- - TLS en tránsito, cifrado en reposo
- - Cumplimiento GDPR (residencia de datos) simple ofuscacion de data sensible

Conclusiones

- ✓ Arquitectura distribuida, modular y escalable
- ✓ Preparada para alta concurrencia (1K jugadores iniciales, escalable)
- ✓ Baja latencia con WebSockets
- ✓ Seguridad en identidad y anti-cheat
- ✓ Costos optimizados con infra open source