**Servidor de juegos de Casino, Póker, Blackjack, Slots en línea impulsado por Redis, node.js y socket.io.  
Desarrollado bajo la licencia de MIT adaptado por Daniel Crucianelli.  
  
 C:\Users\Usuario\Desktop\logo_10_13.png  
  
 Juegos compatibles:**

* ** Deportivas**
* ** Ruleta**
* ** Texas Holdem**
* ** Slots**
* ** Blackjack**

**Características:**

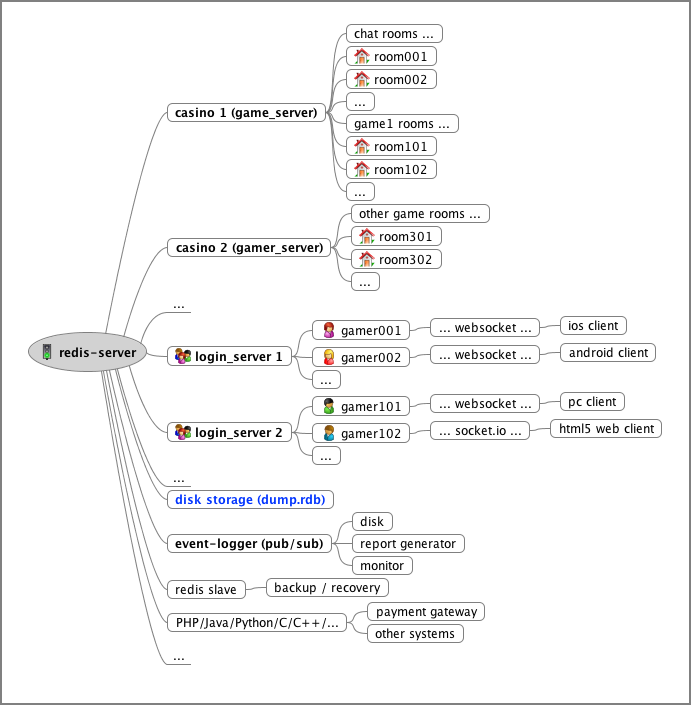
* ** Multiplataforma: impulsado por node.js, fácil de implementar en varias plataformas.**
* ** Escalable: usando Redis como canal de mensajes y almacenamiento de datos.**
* ** Arquitectura abierta: con Redis como canal de mensajes, fácil de interactuar y ampliar.**
* ** Clúster: usando PM2, sesión permanente y socket.io-redis.**
* ** Protocolo WebSocket: viene con API de cliente javascript y demostración basada en web.**
* ** Registrador de eventos para eventos del servidor y acciones del usuario.**

**C:\Users\Usuario\Desktop\logo_10_13.png  
  
  
 C:\Users\Usuario\Desktop\logo_10_13.png  
 Tareas a completar en el proceso de apertura:**

** API de cliente para Unity3D (C #, compatible con mono / .NET2.0)**

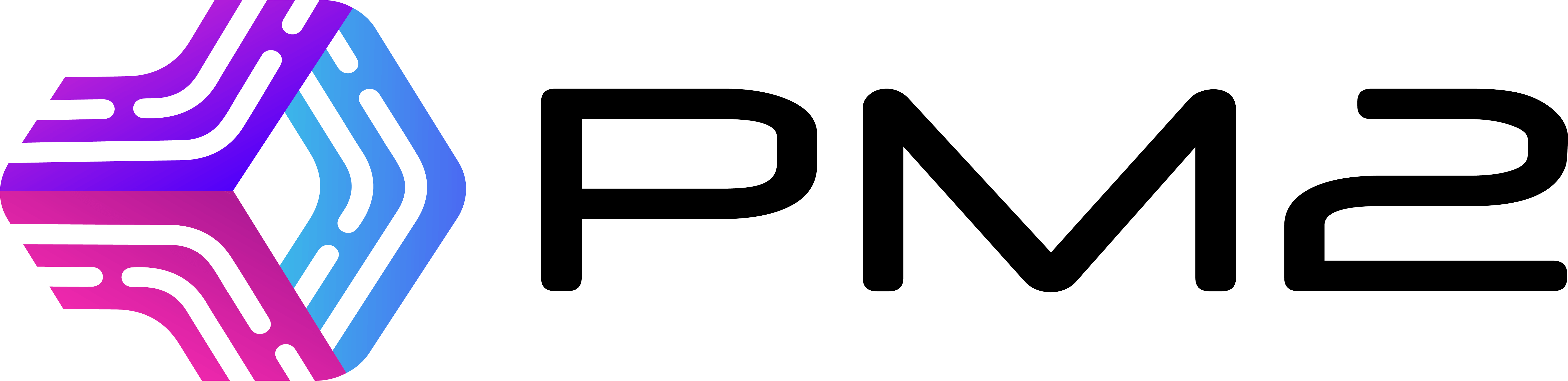
** URL de devolución de llamada de la pasarela de pago.**

** Portal de administración.**

** Equilibrio de carga: usando NginX, o manejado automáticamente si se implementa en Amazon Cloud.  
  
  
ARQUITECTURA DEL PROYECTO: **

## C:\Users\Usuario\Desktop\logo_10_13.png Requerido

* **Redis:**
* **Redis es un almacenamiento y caché de clave-valor avanzados de código abierto, con licencia BSD. A menudo se lo denomina servidor de estructura de datos, ya que las claves pueden contener cadenas, hashes, listas, conjuntos, conjuntos ordenados, mapas de bits e hiperloglogs.**
* [**Descargar**](http://redis.io/download)**: https://redis.io/download**
* **node.js / npm:**
* **Node.js es una plataforma basada en el tiempo de ejecución de JavaScript de Chrome para crear fácilmente aplicaciones de red rápidas y escalables. Node.js utiliza un modelo de E / S sin bloqueo controlado por eventos que lo hace liviano y eficiente, perfecto para aplicaciones en tiempo real con uso intensivo de datos que se ejecutan en dispositivos distribuidos.**
* [**Descargar**](http://nodejs.org/download)**: https://nodejs.org/download/**
* **PM2 (producción y clúster):**

**PM2 es un administrador de procesos de producción para aplicaciones Node.js con un balanceador de carga incorporado. Le permite mantener las aplicaciones activas para siempre, recargarlas sin tiempo de inactividad y facilitar las tareas comunes de administración del sistema.  
Panel:** [**https://id.keymetrics.io/api/oauth/login**](https://id.keymetrics.io/api/oauth/login) ****

* **Iniciar una aplicación en modo de producción es tan fácil como:**
* **$ pm2 iniciar app.js**
* **PM2 es constantemente atacado por más de 1800 pruebas .**
* **Sitio web oficial:**[**https://pm2.keymetrics.io/**](https://pm2.keymetrics.io/)
* **Funciona en Linux (estable) y macOS (estable) y Windows (estable). Todas las versiones de Node.js son compatibles a partir de Node.js 12.X.**

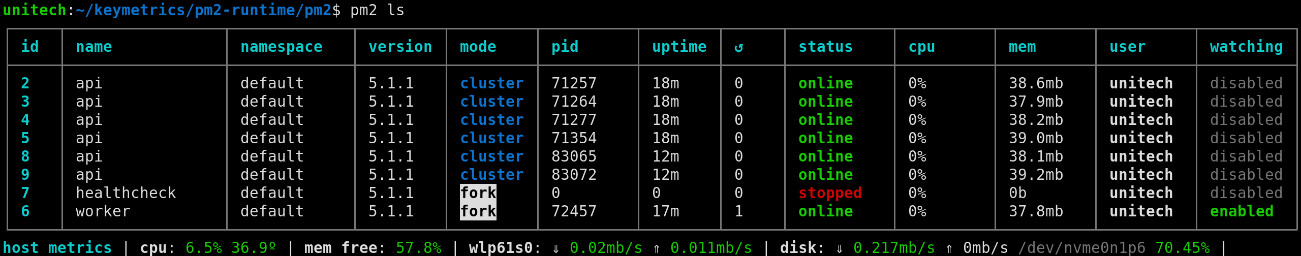
### **Instalación de PM2**

* **Con NPM:**
* **$ npm instalar pm2 -g**
* **Puede instalar Node.js fácilmente con NVM o ASDF .**

### **Iniciar una aplicación**

* **Puede iniciar cualquier aplicación (Node.js, Python, Ruby, binarios en $ PATH ...) así:**
* **$ pm2 iniciar app.js**
* **Su aplicación ahora está demonizada, monitoreada y se mantiene viva para siempre.**

### **Administrar aplicaciones**

* **Una vez iniciadas las aplicaciones, puede administrarlas fácilmente:  
    
  Para enumerar todas las aplicaciones en ejecución:**
* **$ pm2 lista**
* **La gestión de aplicaciones es sencilla:**
* **$ pm2 stop < nombre\_aplicación | espacio de nombres | id | ' todos ' | json\_conf >**
* **$ pm2 reiniciar < nombre\_aplicación | espacio de nombres | id | ' todos ' | json\_conf >**
* **$ pm2 eliminar < nombre\_aplicación | espacio de nombres | id | ' todos ' | json\_conf >**
* **Para tener más detalles sobre una aplicación específica:**
* **$ pm2 describe < id | nombre\_aplicación >**
* **Para monitorear registros, métricas personalizadas, información de la aplicación:**
* **$ pm2 monit**

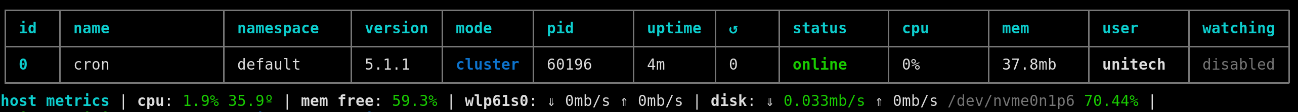
### **Modo de clúster: equilibrio de carga de Node.js y recarga sin tiempo de inactividad**

* **El modo Clúster es un modo especial cuando se inicia una aplicación Node.js, inicia múltiples procesos y equilibra la carga de consultas HTTP / TCP / UDP entre ellos. Esto aumenta el rendimiento general (en un factor de x10 en máquinas de 16 núcleos) y la confiabilidad (reequilibrio de socket más rápido en caso de errores no manejados).  
  **
* **Iniciar una aplicación Node.js en modo de clúster que aprovechará todas las CPU disponibles:**
* **$ pm2 start api.js -i < procesos >**
* **<processes>puede ser 'max', -1(todas las CPU menos 1) o un número específico de instancias para comenzar.**
* **Recarga sin tiempo de inactividad**
* **Hot Reload permite actualizar una aplicación sin ningún tiempo de inactividad:**
* **$ pm2 recargar todo**

### **Soporte para contenedores**

* **Con el comando de reemplazo directo para node, llamado pm2-runtime, ejecute su aplicación Node.js en un entorno de producción reforzado. Usarlo es perfecto:**
* **RUN npm install pm2 -g**
* **CMD [ "pm2-runtime", "npm", "--", "start" ]**

### **Barra de velocidad de monitoreo de host**

* **PM2 permite monitorear los signos vitales de su host / servidor con una barra de velocidad de monitoreo.**
* **Para habilitar la supervisión del host:**
* **$ pm2 establece pm2: sysmonit true**
* **$ pm2 actualización**
* ****

### **Monitoreo basado en terminales C:\Users\Usuario\Desktop\pm2-monit.png**

* **Supervise todos los procesos iniciados directamente desde la línea de comando:**
* **$ pm2 monit**

### **Gestión de registros**

* **Para consultar los registros, simplemente escriba el comando:**
* **$ pm2 registros**
* **Están disponibles las salidas estándar, sin formato, JSON y formateadas.**
* **Ejemplos:**
* **$ pm2 logs APP-NAME # Mostrar los registros de APP-NAME**
* **$ pm2 logs --json # Salida JSON**
* **$ pm2 logs --format # Salida formateada**
* **$ pm2 flush # Vaciar todos los registros**
* **$ pm2 reloadLogs # Recargar todos los registros**
* **Para habilitar la rotación de registros, instale el siguiente módulo**
* **$ pm2 instalar pm2-logrotate**

### **Generación de scripts de inicio**

* **PM2 puede generar y configurar una secuencia de comandos de inicio para mantener activos a PM2 y sus procesos en cada reinicio del servidor.**
* **Sistemas de inicio compatibles: systemd , upstart , launchd , rc.d**
* **# Generar secuencia de comandos de inicio**
* **$ pm2 inicio**
* **# Congele su lista de procesos al reiniciar el servidor**
* **$ pm2 ahorrar**
* **# Eliminar el script de inicio**
* **$ pm2 unstartup**

### **Actualización de PM2**

* **# Instale la última versión de PM2**
* **$ npm instalar pm2 @ latest -g**
* **# Guarde la lista de procesos, salga de PM2 antiguo y restaure todos los procesos**
* **$ pm2 update**
* **Las actualizaciones de PM2 son perfectas**

## Monitoreo PM2 +

* **Si administra sus aplicaciones con PM2, PM2 + facilita el monitoreo y la administración de aplicaciones en los servidores.  
    
    
    
   C:\Users\Usuario\Desktop\logo_10_13.png**

# **C:\Users\Usuario\Desktop\68747470733a2f2f706d322e696f2f696d672f6170702d6f766572766965772e706e67.png Instalación**

* **Instalación global y ejecución como servicio con PM2:**
* **[sudo] npm install pm2 -g**
* **[sudo] npm install casino-server -g**
* **# ejecutar redis-server primero**
* **redis-server &**
* **# Funcionar como servicio de racimo y de modo**
* **PM2 empezar ` que el casino-servidor ` -i 0**
* **Instalar como una aplicación de nodo y ejecutar en la carpeta actual:**
* **Descomprimir archivo zip y ejecutar carpeta actual**
* **cd casino**
* **npm install**
* **# sudo npm install -g gulp**
* **# gulp build**
* **node bin / casino-server [opciones]**
* **Ahora puede visitar http: // localhost: 7000 , es un cliente de juego basado en la web para fines de prueba y demostración.**
* **# abrir un navegador para acceder a la página web de prueba**
* **abrir http: // localhost: 7000 /**

# **Opciones Disponibles:**

* **-p Puerto a usar (predeterminado en 7000)**
* **-h Dirección de host a utilizar (predeterminado en 0.0.0.0)**
* **-r Dirección del servidor Redis (el valor predeterminado es 127.0.0.1:6379)**

# **Instrumentos**

* **Con Redis como el canal de mensajes, es muy fácil para las herramientas trabajar con la arquitectura abierta.  
    
  Registrador de eventos:  
    
  Inicie el registrador de eventos para monitorear los eventos:**
* **bin / logger.js**
* **registre los eventos en el archivo de registro:**
* **bin / logger.js -o ./tmp/casino.log**

# **Ver también**

* **socket.io para Unity3D , es una biblioteca cliente socket.io para Unity3D. Puede desarrollar juegos de casino con Unity3D y utilizar esta biblioteca para conectarse a este servidor de casino.**
* **  
  Biblioteca de cliente Socket.IO para Unity (mono / .NET 2.0), que se transfirió desde la versión 1.1.0 del cliente JavaScript .**
* **SocketIoClientDotNet de Quobject es un proyecto muy bueno, pero no es compatible con Unity. Así que pasé algunas noches para portarlo a mono / .NET 2.0.**
* **Ahora los desarrolladores de juegos pueden:**
* **use node.js para desarrollar servidores de juegos, disfrute del poderoso ecosistema de red y base de datos, implemente fácilmente en la nube;**
* **utilice Unity para desarrollar un cliente de juego, disfrute del potente motor de juego y el IDE.**
* **use websocket / socket.io para comunicarse entre servidor / cliente.**

## Instalación

* **Descargue el paquete socket.io.unity y luego impórtelo en Unity.**
* **O, solo descargue los siguientes archivos en Lib y colóquelos en el proyecto Unity:**
* **WebSocket4Net.dll**
* **SocketIoClientDotNet.dll**
* **Newtonsoft.Json.dll**

## Uso

* **El cliente socket.io para Unity tiene una API similar a las del cliente JavaScript .**
* **utilizando Quobject . SocketIoClientDotNet . Cliente ;**
* **var socket = IO . Socket ( " http: // localhost: 3000 " );**
* **zócalo . Encendido ( Socket . EVENT\_CONNECT , () =>**
* **{**
* **zócalo . Emitir ( " hola " );**
* **});**
* **zócalo . En ( " hola " , ( datos ) =>**
* **{**
* **Depurar . Registro ( datos );**
* **zócalo . Desconectar ();**
* **});**
* **Y, con Newtonsoft.Json.dll, podemos fácilmente serializar / deserializar el objeto json. Leer más sobre Newtonsoft.Json .**

## Características

* **Esta biblioteca admite todas las funciones que hace el cliente JS, incluidos eventos, opciones y transporte de actualización.**

**C:\Users\Usuario\Desktop\logo_10_13.png**