**UNIVERSIDAD NACIONAL DEL ALTIPLANO**

**FACULTAD DE ESTADISTICA E INFORMATICA**

# ESCUELA PROFESIONAL DE INGENIERIA ESTADISTICA E

**INFORMATICA**



**MEMORIA DESCRIPTIVA:**

**AREA: SISTEMAS DISTRIBUIDOS**

**DOCENTE: ING. ZANABRIA TICONA EDSON DENIS**

**CICLO: VII SEMESTRE**

**SECCIÓN: “A”**

**PRESENTADO POR:**

# ARIAS VILCA, Jose David

**CUELA SALAS, Abdiel Fernando**

**PUNO ----- PERÚ**

**2024**

### Introducción

El desarrollo de aplicaciones web ha evolucionado significativamente en la última década, impulsado por la necesidad de crear plataformas más interactivas, eficientes y escalables. En este contexto, la elección de tecnologías y frameworks adecuados es crucial para el éxito del proyecto. En este informe, presentamos el desarrollo de una aplicación web diseñada como un repositorio de juegos retro, utilizando el framework CodeIgniter 3, implementada en un servidor de Azure con la pila LAMP (Linux, Apache, MySQL, PHP). La dirección IP de nuestro servidor es 52.188.195.72.

#### Objetivo del Proyecto

El objetivo principal de este proyecto es proporcionar una plataforma donde los usuarios puedan buscar y obtener información detallada sobre una amplia gama de juegos retro. Esta plataforma debe ser intuitiva y accesible, permitiendo a los usuarios disfrutar de una experiencia de usuario fluida desde múltiples dispositivos. Además, buscamos fomentar la interacción entre los usuarios mediante un chat en tiempo real y ofrecer un acceso flexible a los datos a través de APIs REST.

#### Elección de CodeIgniter 3

**1. Sencillez y Velocidad de Desarrollo:** CodeIgniter 3 es un framework ligero y fácil de usar que sigue el patrón de arquitectura MVC (Modelo-Vista-Controlador). Su sencillez permite a los desarrolladores crear aplicaciones rápidamente sin la necesidad de una configuración compleja. Además, su documentación extensa y clara facilita la curva de aprendizaje, permitiendo a los desarrolladores nuevos familiarizarse rápidamente con el framework.

**2. Flexibilidad y Escalabilidad:** A pesar de su ligereza, CodeIgniter es extremadamente flexible y escalable. Permite la integración de diversas bibliotecas y herramientas según las necesidades del proyecto. Su estructura modular facilita el mantenimiento y la ampliación de la aplicación, asegurando que la plataforma pueda crecer junto con la base de usuarios y las demandas futuras.

**3. Seguridad Integrada:** CodeIgniter incluye múltiples características de seguridad integradas, como la protección contra ataques CSRF (Cross-Site Request Forgery) y XSS (Cross-Site Scripting). Además, proporciona funciones de encriptación para manejar datos sensibles, como contraseñas de usuarios. Estas características aseguran que la aplicación sea resistente a las amenazas comunes de seguridad web.

#### Implementación en Servidor Azure con LAMP

**1. Fiabilidad y Escalabilidad de Azure:** Azure es una de las plataformas en la nube más confiables y escalables disponibles en la actualidad. Ofrece una amplia gama de servicios que permiten la implementación, gestión y monitoreo de aplicaciones de manera eficiente. Al elegir Azure, garantizamos que nuestra aplicación pueda manejar grandes volúmenes de tráfico y escalar según sea necesario.

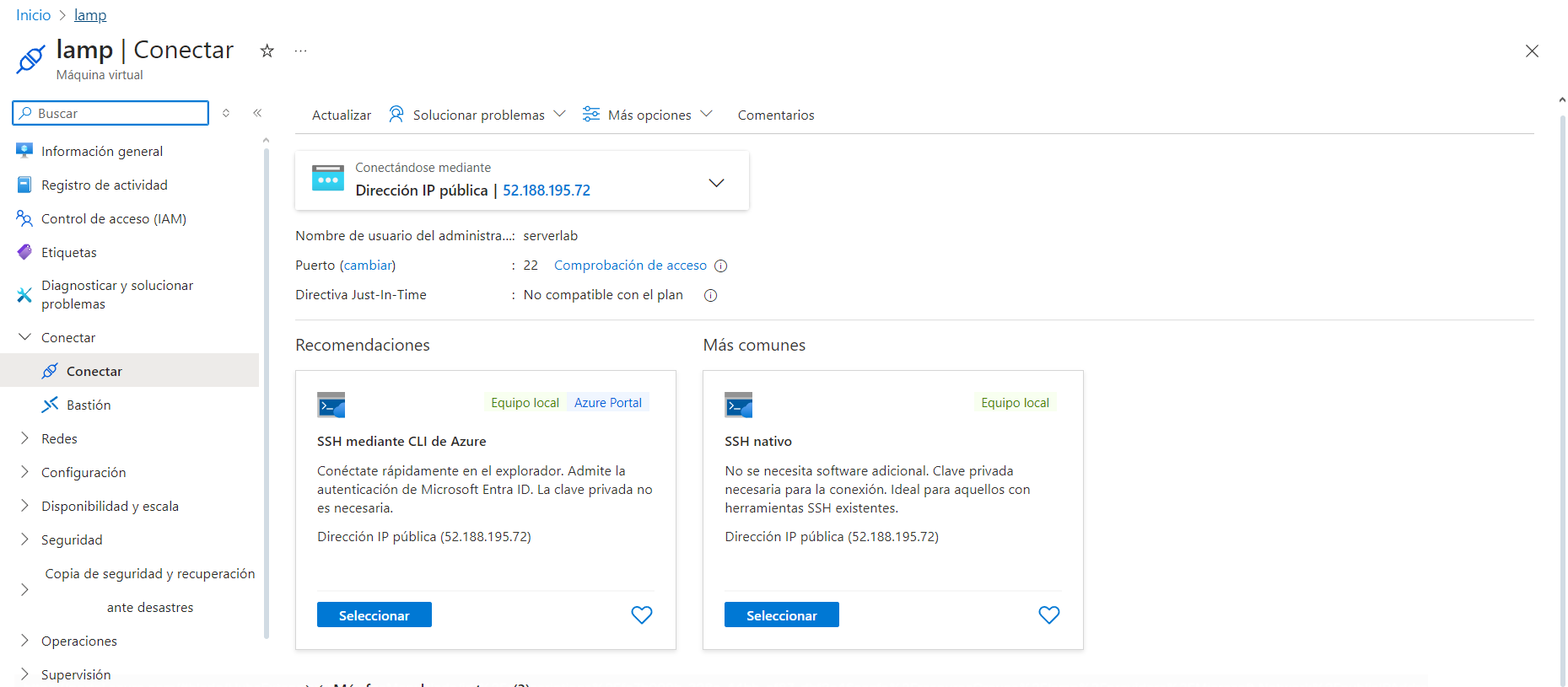
**2. Pila LAMP:** La pila LAMP (Linux, Apache, MySQL, PHP) es una de las configuraciones más populares y probadas para el desarrollo web. Linux proporciona un entorno de servidor seguro y estable, Apache actúa como un servidor web robusto y flexible, MySQL maneja la gestión de bases de datos de manera eficiente, y PHP es un lenguaje de scripting potente y ampliamente utilizado. Esta combinación asegura que nuestra aplicación sea fiable y pueda manejar el tráfico de manera eficiente.

#### Justificación del Uso de Node.js con Socket.IO

**1. Comunicación en Tiempo Real:** Node.js, combinado con Socket.IO, es ideal para manejar aplicaciones que requieren comunicación en tiempo real. En nuestra aplicación, esto es crucial para la funcionalidad del chat, permitiendo a los usuarios interactuar entre sí de manera instantánea. Socket.IO facilita la creación de conexiones persistentes entre el cliente y el servidor, asegurando que los mensajes se transmitan con baja latencia.

**2. Rendimiento y Escalabilidad:** Node.js es conocido por su capacidad de manejar un gran número de conexiones concurrentes con baja latencia y alta eficiencia. Esto es posible gracias a su modelo de ejecución basado en eventos y su uso de V8, el motor de JavaScript de alto rendimiento de Google. Al utilizar Node.js para el chat, garantizamos que la aplicación pueda manejar múltiples usuarios interactuando en tiempo real sin comprometer el rendimiento.

#### Imagen de servidor Azure con Ubuntu server: teniendo el lamp ya instalado



### Arquitectura del Sistema

**2.1 Descripción General** La arquitectura del sistema para el repositorio de juegos retro está diseñada para ser modular, escalable y eficiente. Este diseño facilita el mantenimiento y la escalabilidad de la aplicación, permitiendo integrar nuevas funcionalidades a medida que crece la base de usuarios. La arquitectura consta de los siguientes componentes principales:

* **Servidor Web (LAMP en Azure):**
  + **Linux:** El sistema operativo que proporciona una plataforma estable y segura.
  + **Apache:** El servidor web que maneja las solicitudes HTTP y entrega los recursos estáticos (HTML, CSS, JavaScript e imágenes).
  + **MySQL:** El sistema de gestión de bases de datos que almacena toda la información de los juegos, usuarios y otros datos de la aplicación.
  + **PHP:** El lenguaje de programación del lado del servidor, usado para construir la lógica de la aplicación con el framework CodeIgniter 3.
* **Framework CodeIgniter 3:**
  + **Modelo-Vista-Controlador (MVC):** CodeIgniter sigue el patrón MVC, que separa la lógica de la aplicación (Modelos), la presentación (Vistas) y el manejo de las solicitudes (Controladores). Esto facilita el desarrollo y mantenimiento de la aplicación.
  + **Bibliotecas y Helpers:** CodeIgniter proporciona una serie de bibliotecas y helpers integrados que aceleran el desarrollo y proporcionan funcionalidad común (como manejo de formularios, sesiones, validaciones, etc.).
* **Servidor de Socket (Node.js con Socket.IO):**
  + **Node.js:** Utilizado para construir el servidor que maneja la comunicación en tiempo real. Node.js es conocido por su capacidad de manejar múltiples conexiones concurrentes de manera eficiente.
  + **Socket.IO:** Una biblioteca que facilita la comunicación en tiempo real entre el servidor y los clientes. En esta aplicación, se utiliza para implementar la funcionalidad de chat en tiempo real, permitiendo a los usuarios interactuar instantáneamente.
* **APIs REST:**
  + Las APIs REST permiten la interacción programática con los datos del repositorio de juegos. Los desarrolladores externos pueden usar estas APIs para acceder y manipular los datos de los juegos, facilitando integraciones con otras aplicaciones y servicios.

**2.2 Diagrama de Arquitectura**

+-------------------+ +-------------------+

| Cliente Web | | Servidor |

+-------------------+ +-------------------+

| Navegador (UI) | <--------> | CodeIgniter (MVC)|

+-------------------+ +-------------------+

^ |

| v

+-------------------+ +-------------------+

| Cliente Socket | <--------> | Servidor Socket |

+-------------------+ | (Node.js) |

^ /| +-----------------+|

| / | Base de Datos |

v / +-------------------+

+-------------------+

| APIs REST |

+-------------------+

**2.3 Detalles de la Implementación en Azure** El servidor de Azure proporciona un entorno robusto y escalable para nuestra aplicación. La implementación en Azure asegura que nuestra aplicación pueda manejar grandes volúmenes de tráfico y puede escalar horizontalmente cuando sea necesario.

* **Máquinas Virtuales (VMs):** Utilizamos máquinas virtuales en Azure para ejecutar la pila LAMP. Estas VMs están configuradas para autoescalado, permitiendo incrementar los recursos en respuesta a la demanda del tráfico.
* **Balanceador de Carga (Load Balancer):** Azure Load Balancer distribuye las solicitudes entrantes entre múltiples instancias de la aplicación, asegurando alta disponibilidad y rendimiento.
* **Copias de Seguridad y Recuperación ante Desastres:** Azure proporciona servicios de copia de seguridad y recuperación ante desastres, asegurando la integridad y disponibilidad de los datos.

funcionalidades

### Funcionalidades

La aplicación de repositorio de juegos retro implementada en CodeIgniter 3 ofrece una variedad de funcionalidades diseñadas para proporcionar una experiencia de usuario rica e interactiva. A continuación, se detallan las principales características de la aplicación:

**3.1 Middleware de Autenticación**

* **Función:** Asegura que solo los usuarios autenticados puedan acceder a ciertas rutas y funcionalidades de la aplicación.
* **Implementación:** Utiliza una clase middleware personalizada en CodeIgniter que verifica la sesión del usuario antes de permitir el acceso a las rutas protegidas. Esta clase se integra en el sistema de enrutamiento de CodeIgniter, garantizando que todas las solicitudes pasen por este filtro de seguridad antes de ser procesadas.
* **Beneficio:** Mejora la seguridad de la aplicación al restringir el acceso a usuarios no autorizados, protegiendo los datos sensibles y las funcionalidades críticas.

**3.2 Chat en Tiempo Real**

* **Función:** Permite la comunicación en tiempo real entre los usuarios mediante un sistema de chat.
* **Implementación:** Utiliza Node.js y Socket.IO para manejar las conexiones en tiempo real. El servidor de Node.js escucha las conexiones entrantes y gestiona la transmisión de mensajes entre los usuarios conectados. El cliente del chat está integrado en la vista de CodeIgniter, utilizando JavaScript para interactuar con el servidor de Socket.IO.
* **Características Adicionales:** Soporte para mensajes de texto, notificaciones en tiempo real, y la capacidad de unirse a diferentes salas de chat.
* **Beneficio:** Fomenta la interacción y la comunidad entre los usuarios del repositorio de juegos, mejorando la experiencia del usuario.

**3.3 API REST**

* **Función:** Proporciona un mecanismo para la gestión y consumo de datos de la aplicación a través de solicitudes HTTP.
* **Implementación:** Se crean controladores en CodeIgniter para manejar las solicitudes API, y se utilizan para obtener y enviar datos en formato JSON. Cada endpoint de la API está diseñado siguiendo los principios RESTful, asegurando una comunicación coherente y predecible.
* **Operaciones Soportadas:** CRUD (Crear, Leer, Actualizar, Eliminar) sobre los datos de los juegos y usuarios.
* **Beneficio:** Facilita la integración de la aplicación con otros servicios y aplicaciones externas, permitiendo a los desarrolladores externos acceder y manipular los datos del repositorio de juegos.

**3.4 Información de Juegos**

* **Función:** Permite a los usuarios buscar y ver información detallada sobre juegos retro.
* **Implementación:** Las vistas muestran los detalles de cada juego, incluyendo su descripción, año de lanzamiento, desarrollador, género, y más. Los datos se obtienen de la base de datos y se presentan de manera clara y accesible, utilizando plantillas HTML y estilos CSS.
* **Características Adicionales:** Filtros avanzados de búsqueda, categorización de juegos por género y año, y la capacidad de añadir juegos a una lista de favoritos.
* **Beneficio:** Proporciona una base de datos rica y bien organizada de juegos retro, facilitando la búsqueda y consulta de información por parte de los usuarios.

**3.5 Administración de Usuarios**

* **Función:** Permite la gestión de cuentas de usuario, incluyendo el registro, inicio de sesión y recuperación de contraseñas.
* **Implementación:** Formularios de autenticación personalizados y mecanismos de validación del lado del servidor para asegurar la integridad de los datos.
* **Características Adicionales:** Perfil de usuario, historial de juegos consultados y estadísticas de uso.
* **Beneficio:** Mejora la experiencia del usuario al proporcionar un sistema de gestión de cuentas fácil de usar y seguro.

**3.6 Gestión de Contenidos**

* **Función:** Permite a los administradores agregar, editar y eliminar juegos del repositorio.
* **Implementación:** Panel de administración con formularios intuitivos para la gestión de contenidos.
* **Beneficio:** Facilita la actualización y mantenimiento de la base de datos de juegos, asegurando que la información esté siempre actualizada y relevante.

Estas funcionalidades se combinan para ofrecer una experiencia de usuario completa y satisfactoria, permitiendo a los usuarios interactuar con el repositorio de juegos de diversas maneras y obteniendo el máximo valor de la aplicación.

#### API rest se aplica el d.n.i y muestra el “jugador “sus características como los juegos que le gusta



### *API que muestra la información de juegos* **RAWG Video Games Database** Una API completa que proporciona datos sobre juegos, plataformas, géneros y más. Puedes obtener información detallada sobre juegos, como fechas de lanzamiento, puntuaciones, descripciones, etc. Enlace a RAWG API.

### 

### Seguridad

La seguridad es un aspecto crítico en el desarrollo de cualquier aplicación web. En el caso del repositorio de juegos retro implementado en CodeIgniter 3, se han tomado varias medidas para asegurar que la aplicación sea robusta frente a diversas amenazas de seguridad. A continuación, se detallan las estrategias y técnicas utilizadas para garantizar la seguridad de la aplicación y de los datos de los usuarios:

**4.1 Autenticación de Usuarios**

* **Middleware de Autenticación:**
  + **Función:** Asegura que solo los usuarios autenticados puedan acceder a ciertas rutas y funcionalidades de la aplicación.
  + **Implementación:** Utilizamos una clase middleware personalizada en CodeIgniter que verifica la sesión del usuario antes de permitir el acceso a las rutas protegidas. Si un usuario no está autenticado, el middleware redirige la solicitud a la página de inicio de sesión.
  + **Beneficio:** Mejora la seguridad de la aplicación al restringir el acceso a usuarios no autorizados, protegiendo los datos sensibles y las funcionalidades críticas.

**4.2 Gestión Segura de Contraseñas**

* **Almacenamiento de Contraseñas:**
  + **Función:** Las contraseñas de los usuarios se almacenan de manera segura utilizando algoritmos de hash y sal.
  + **Implementación:** Utilizamos la función password\_hash de PHP para encriptar las contraseñas antes de almacenarlas en la base de datos. Este algoritmo genera un hash irreversible de la contraseña, lo que significa que incluso si la base de datos es comprometida, las contraseñas no pueden ser revertidas a su forma original.
  + **Beneficio:** Asegura que las contraseñas de los usuarios estén protegidas contra ataques de fuerza bruta y de diccionario.

**4.3 Comunicación Segura**

* **SSL/TLS:**
  + **Función:** Asegura que las comunicaciones entre el cliente y el servidor sean cifradas y protegidas contra interceptaciones.
  + **Implementación:** Se recomienda implementar certificados SSL/TLS en el servidor web para cifrar las conexiones HTTPS. Esto asegura que los datos transmitidos entre el navegador del usuario y el servidor estén protegidos contra ataques de interceptación (man-in-the-middle).
  + **Beneficio:** Protege la integridad y confidencialidad de los datos durante la transmisión, aumentando la confianza de los usuarios en la plataforma.

**4.4 Protección de APIs**

* **Tokens de Acceso:**
  + **Función:** Asegura que solo usuarios autorizados puedan realizar solicitudes a las APIs REST.
  + **Implementación:** Utilizamos tokens de acceso que se generan al momento de la autenticación del usuario. Estos tokens deben ser enviados en cada solicitud a las APIs para verificar la identidad del usuario. Los tokens tienen una vida útil limitada y pueden ser revocados si se detecta actividad sospechosa.
  + **Beneficio:** Protege las APIs contra el acceso no autorizado y asegura que solo los usuarios legítimos puedan interactuar con los datos sensibles.

**4.5 Prevención de Ataques Comunes**

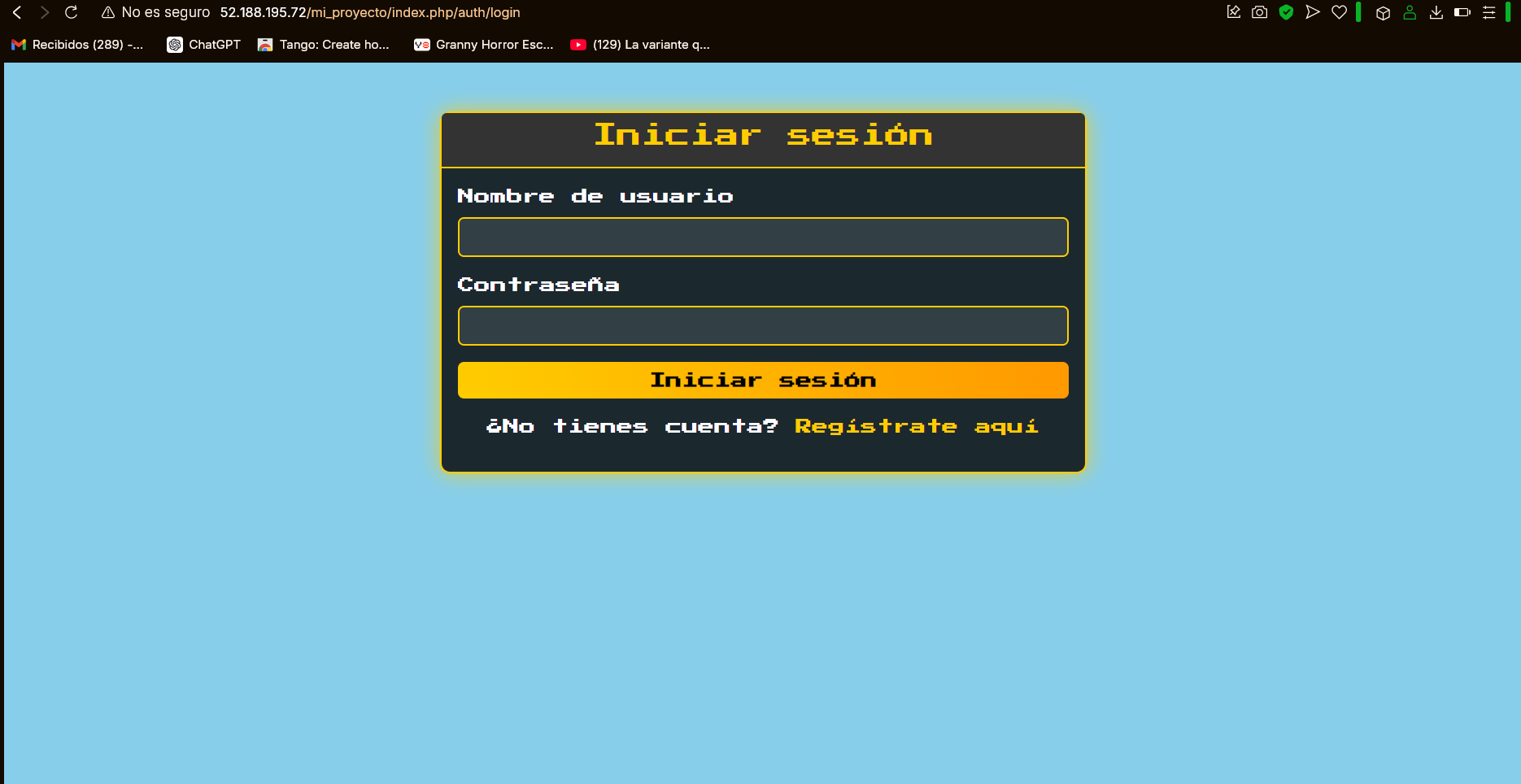
* **Protección contra CSRF (Cross-Site Request Forgery):**
  + **Función:** Previene que atacantes ejecuten acciones no autorizadas en nombre de los usuarios autenticados.
  + **Implementación:** CodeIgniter tiene soporte integrado para protección CSRF, que se habilita a través de su configuración. Esta protección genera un token único para cada solicitud de formulario, que debe ser validado por el servidor.
  + **Beneficio:** Protege la aplicación contra solicitudes maliciosas que intentan aprovecharse de la sesión activa del usuario.
* **Protección contra XSS (Cross-Site Scripting):**
  + **Función:** Previene la inyección de código malicioso en las páginas web que podría ser ejecutado por los navegadores de los usuarios.
  + **Implementación:** Utilizamos las funciones de escape de salida de CodeIgniter para limpiar todos los datos de entrada y salida, evitando la inyección de scripts maliciosos.
  + **Beneficio:** Protege a los usuarios de ataques que podrían robar información sensible o comprometer su experiencia de navegación.

**4.6 Auditoría y Monitoreo**

* **Registros de Actividad (Logs):**
  + **Función:** Monitorea y registra las actividades del sistema y de los usuarios para detectar y responder a actividades sospechosas.
  + **Implementación:** CodeIgniter proporciona un sistema de logging que permite registrar eventos importantes, como intentos de inicio de sesión, errores del sistema y cambios en la configuración.
  + **Beneficio:** Facilita la detección de patrones de comportamiento inusuales y permite una respuesta rápida a incidentes de seguridad.
* **Herramientas de Monitoreo:**
  + **Función:** Monitorea el rendimiento y la seguridad de la aplicación en tiempo real.
  + **Implementación:** Utilizamos herramientas de monitoreo proporcionadas por Azure, como Azure Monitor y Azure Security Center, para supervisar el estado de la aplicación, detectar vulnerabilidades y recibir alertas en caso de incidentes de seguridad.
  + **Beneficio:** Asegura que cualquier problema de seguridad o rendimiento sea identificado y abordado de manera proactiva.

Estas estrategias y medidas de seguridad se combinan para asegurar que el repositorio de juegos retro sea una plataforma segura y confiable para todos los usuarios, protegiendo tanto los datos sensibles como la integridad de la aplicación.

#### Imagen de login con el estilo el cual nos pide el usuario y contraseña para iniciar sesión



#### Imagen de registro de usuario con contraseña encriptada

### Interfaz de Usuario

La interfaz de usuario (UI) es un componente crucial del repositorio de juegos retro, ya que determina cómo los usuarios interactúan con la aplicación y acceden a la información de los juegos. Una interfaz bien diseñada no solo mejora la experiencia del usuario, sino que también facilita la navegación y la accesibilidad. A continuación, se detallan los aspectos clave de la UI de la aplicación:

**5.1 Diseño Responsivo**

* **Adaptabilidad:** La interfaz está diseñada para ser completamente responsiva, asegurando que se vea y funcione bien en una variedad de dispositivos, incluyendo smartphones, tablets y computadoras de escritorio. Utilizamos técnicas de diseño responsivo como media queries y layouts flexibles para adaptar el contenido según el tamaño de la pantalla.
* **Beneficio:** Los usuarios pueden acceder al repositorio de juegos desde cualquier dispositivo, disfrutando de una experiencia coherente y optimizada independientemente del medio que utilicen.

**5.2 Navegación Intuitiva**

* **Disposición de Elementos:** La disposición de los elementos de la UI está pensada para ser intuitiva y fácil de usar. Los menús de navegación son claros y accesibles, permitiendo a los usuarios encontrar rápidamente lo que buscan. Utilizamos iconos y etiquetas descriptivas para mejorar la usabilidad.
* **Flujo de Usuario:** Los flujos de usuario están diseñados para ser lógicos y sencillos. Desde la página de inicio hasta la búsqueda de juegos y la interacción en el chat, cada acción está pensada para ser intuitiva y directa.
* **Beneficio:** Facilita la interacción del usuario con la aplicación, reduciendo la curva de aprendizaje y mejorando la satisfacción general.

**5.3 Componentes Principales**

* **Formularios de Autenticación:**
  + **Registro e Inicio de Sesión:** Formularios sencillos y seguros para que los usuarios se registren y accedan a la plataforma. Se incluyen validaciones tanto del lado del cliente como del servidor para garantizar la integridad de los datos.
  + **Recuperación de Contraseña:** Mecanismo para que los usuarios recuperen sus contraseñas en caso de olvido, enviando un enlace seguro a su correo electrónico registrado.
* **Panel de Chat:**
  + **Comunicación en Tiempo Real:** Zona interactiva donde los usuarios pueden enviar y recibir mensajes en tiempo real. La UI del chat incluye notificaciones de nuevos mensajes y actualizaciones en tiempo real sin necesidad de refrescar la página.
  + **Salas de Chat:** Los usuarios pueden unirse a diferentes salas de chat temáticas, facilitando la conversación sobre temas específicos relacionados con los juegos.
* **Dashboards:**
  + **Visualización de Datos:** Los dashboards ofrecen una vista general de la actividad del usuario, incluyendo estadísticas de juegos consultados, comentarios realizados y juegos favoritos. Los datos se presentan de manera clara y visual, utilizando gráficos y tablas cuando es necesario.
  + **Gestión de la Cuenta:** Los usuarios pueden actualizar su información personal, cambiar su contraseña y gestionar sus preferencias desde el panel de control.
* **Buscador de Juegos:**
  + **Barra de Búsqueda:** Una barra de búsqueda potente que permite a los usuarios encontrar rápidamente juegos específicos. Implementa autocompletado y sugerencias en tiempo real para mejorar la experiencia de búsqueda.
  + **Resultados Detallados:** Los resultados de la búsqueda muestran información relevante de cada juego, incluyendo su descripción, género, desarrollador y fecha de lanzamiento. Los juegos también pueden filtrarse por diferentes criterios como género y año de lanzamiento.

**5.4 Estética y Usabilidad**

* **Estilo Visual:**
  + **Tema Retro:** La interfaz utiliza un estilo visual que refleja la temática retro de los juegos, incorporando colores y elementos gráficos que evocan la nostalgia de los videojuegos clásicos.
  + **Consistencia:** Se mantiene una estética consistente en toda la aplicación, utilizando una paleta de colores y tipografía uniforme para asegurar una experiencia visual coherente.
* **Interactividad:**
  + **Feedback Inmediato:** La UI proporciona feedback inmediato a las acciones del usuario, utilizando animaciones y cambios de estado para indicar interacciones exitosas o errores.
  + **Accesibilidad:** Se asegura que la aplicación sea accesible para usuarios con discapacidades, siguiendo las directrices de accesibilidad web (WCAG). Esto incluye soporte para lectores de pantalla y navegación por teclado.

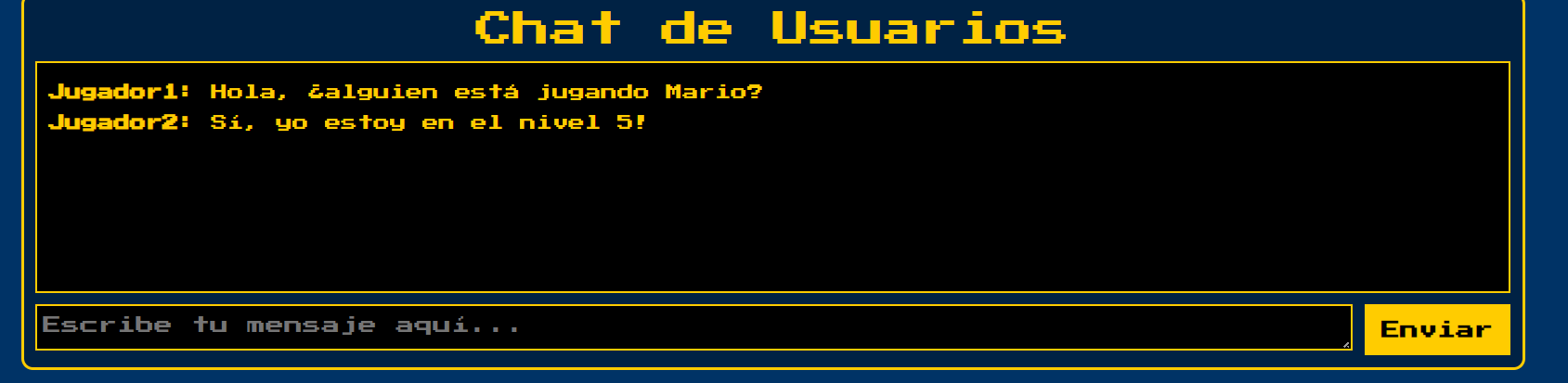
**5.5 Mejora Continua**

* **Pruebas de Usabilidad:** Se realizan pruebas de usabilidad periódicas para identificar posibles problemas y mejorar la UI. Los comentarios de los usuarios son esenciales para realizar ajustes y optimizaciones.
* **Actualizaciones de Diseño:** La interfaz de usuario se actualiza regularmente para incorporar mejoras y nuevas funcionalidades, asegurando que la aplicación evolucione junto con las necesidades de los usuarios.

La interfaz de usuario está diseñada para ofrecer una experiencia de uso amigable, atractiva y eficiente, alineada con la temática retro de los juegos y las expectativas de los usuarios modernos.



### Permitir la comunicación entre los usuarios, ya sea para discutir sobre los juegos, compartir información o para organizar partidas en línea Mantenimiento.



El mantenimiento de la aplicación es esencial para asegurar su funcionamiento continuo, la seguridad y la satisfacción del usuario. A continuación, se detallan las prácticas y procedimientos recomendados para el mantenimiento del repositorio de juegos retro implementado en CodeIgniter 3.

**6.1 Actualizaciones del Sistema**

* **Framework y Librerías:**
  + **Función:** Mantener el framework CodeIgniter y las librerías de Node.js actualizadas a sus versiones más recientes para asegurar la compatibilidad y seguridad.
  + **Procedimiento:** Revisión periódica de las actualizaciones disponibles para CodeIgniter y Node.js. Las actualizaciones pueden incluir mejoras de rendimiento, nuevas funcionalidades y parches de seguridad. Implementar un proceso de prueba antes de aplicar las actualizaciones al entorno de producción para asegurar que no se introduzcan errores o incompatibilidades.
  + **Beneficio:** Las actualizaciones regulares ayudan a proteger la aplicación contra vulnerabilidades conocidas y a mejorar el rendimiento general.

**6.2 Monitoreo del Sistema**

* **Logs de Actividad:**
  + **Función:** Implementar el registro de logs para monitorear el rendimiento y detectar posibles errores.
  + **Procedimiento:** Utilizar las capacidades de logging de CodeIgniter y Node.js para registrar eventos clave, como errores del sistema, intentos de inicio de sesión y cambios en la configuración. Analizar regularmente estos logs para identificar patrones anómalos o actividades sospechosas.
  + **Beneficio:** Los logs detallados permiten a los administradores identificar problemas rápidamente y resolverlos antes de que afecten a los usuarios.
* **Health Checks:**
  + **Función:** Realizar comprobaciones periódicas de salud del sistema para asegurar que todos los servicios estén funcionando correctamente.
  + **Procedimiento:** Implementar scripts de monitorización que verifiquen la disponibilidad y el rendimiento de los componentes críticos de la aplicación (servidor web, base de datos, APIs, etc.). Utilizar herramientas de monitoreo de Azure, como Azure Monitor y Azure Security Center, para recibir alertas en tiempo real.
  + **Beneficio:** Asegura que cualquier problema de rendimiento o disponibilidad sea identificado y abordado de manera proactiva.

**6.3 Gestión de la Base de Datos**

* **Mantenimiento de la Base de Datos:**
  + **Función:** Asegurar que la base de datos esté optimizada y libre de errores.
  + **Procedimiento:** Realizar tareas regulares de mantenimiento de la base de datos, como la optimización de índices, limpieza de datos obsoletos y comprobación de la integridad de los datos. Programar backups periódicos para asegurar que los datos puedan ser recuperados en caso de falla.
  + **Beneficio:** Mantiene el rendimiento de la base de datos y asegura la integridad y disponibilidad de los datos.

**6.4 Seguridad**

* **Auditorías de Seguridad:**
  + **Función:** Realizar auditorías de seguridad regulares para identificar y mitigar vulnerabilidades.
  + **Procedimiento:** Programar auditorías internas y externas para revisar la configuración del servidor, las prácticas de codificación y los mecanismos de autenticación. Implementar parches y ajustes de configuración basados en los hallazgos de las auditorías.
  + **Beneficio:** Asegura que la aplicación esté protegida contra nuevas amenazas de seguridad y que cumpla con las mejores prácticas de la industria.
* **Actualizaciones de Seguridad:**
  + **Función:** Mantener el sistema operativo y las aplicaciones de terceros actualizadas con los últimos parches de seguridad.
  + **Procedimiento:** Configurar actualizaciones automáticas para el sistema operativo Linux y aplicar manualmente los parches de seguridad para aplicaciones críticas como Apache, MySQL y Node.js. Revisar regularmente los boletines de seguridad y aplicar actualizaciones de inmediato cuando se descubren vulnerabilidades críticas.
  + **Beneficio:** Protege la aplicación contra exploits conocidos y asegura un entorno seguro para los usuarios.

**6.5 Documentación**

* **Código:**
  + **Función:** Asegurarse de que el código esté bien documentado para facilitar futuras modificaciones y mantenimiento.
  + **Procedimiento:** Mantener comentarios y documentación actualizada en el código fuente, explicando las funcionalidades y la lógica de las implementaciones. Utilizar herramientas de documentación automática cuando sea posible.
  + **Beneficio:** La documentación clara y concisa ayuda a otros desarrolladores a entender la lógica y estructura del código, facilitando el mantenimiento y las futuras actualizaciones.
* **Manual de Usuario:**
  + **Función:** Mantener el manual de usuario actualizado con las últimas funcionalidades.
  + **Procedimiento:** Revisar y actualizar el manual de usuario cada vez que se introduzcan nuevas funcionalidades o se realicen cambios significativos en la aplicación. Incluir secciones de preguntas frecuentes (FAQ) y soluciones a problemas comunes.
  + **Beneficio:** Asegura que los usuarios tengan acceso a guías y recursos actualizados, mejorando su experiencia y satisfacción.

Estas prácticas de mantenimiento aseguran que el repositorio de juegos retro siga siendo una plataforma fiable, segura y eficiente, proporcionando una experiencia de usuario óptima y protegiendo los datos de los usuarios.

### Conclusiones

El desarrollo de la aplicación web de repositorio de juegos retro utilizando CodeIgniter 3, implementada en un servidor de Azure con la pila LAMP (Linux, Apache, MySQL, PHP) y Node.js con Socket.IO, ha resultado ser una solución robusta, eficiente y segura. A lo largo del proceso de desarrollo, se han alcanzado varios objetivos clave y se han implementado múltiples funcionalidades esenciales que destacan la versatilidad y capacidad de la aplicación. A continuación, se presentan las conclusiones más relevantes:

**Eficiencia y Rapidez de Desarrollo:**

* La elección de CodeIgniter 3 como framework permitió un desarrollo rápido y eficiente, gracias a su estructura ligera y su enfoque en el patrón MVC (Modelo-Vista-Controlador). La facilidad de uso y la extensa documentación de CodeIgniter aceleraron significativamente el proceso de desarrollo y permitieron la implementación de funcionalidades complejas de manera ágil.

**Escalabilidad y Flexibilidad:**

* La arquitectura modular de la aplicación, combinada con la infraestructura escalable de Azure, asegura que el repositorio de juegos pueda crecer y adaptarse a las necesidades futuras. La capacidad de escalar horizontalmente, junto con el uso de balanceadores de carga, garantiza un rendimiento óptimo incluso en situaciones de alta demanda.

**Seguridad y Protección de Datos:**

* Se implementaron diversas medidas de seguridad para proteger los datos de los usuarios y asegurar la integridad de la aplicación. El uso de middleware para la autenticación, la encriptación de contraseñas y la implementación de SSL/TLS son fundamentales para mantener un entorno seguro. Además, las auditorías de seguridad y el monitoreo continuo proporcionan una capa adicional de protección contra posibles amenazas.

**Experiencia de Usuario:**

* La interfaz de usuario responsiva y bien diseñada mejora significativamente la experiencia del usuario, permitiendo una navegación intuitiva y accesible desde cualquier dispositivo. Las funcionalidades como la búsqueda avanzada de juegos, el panel de chat en tiempo real y los dashboards personalizados, enriquecen la interacción del usuario con la plataforma y fomentan un sentido de comunidad.

**Integración y Extensibilidad:**

* La implementación de APIs REST facilita la integración de la aplicación con otros servicios y aplicaciones externas, permitiendo una mayor flexibilidad y extensibilidad. Esto no solo mejora la funcionalidad de la aplicación, sino que también abre oportunidades para colaboraciones y expansiones futuras.

**Mantenimiento y Sostenibilidad:**

* Las prácticas de mantenimiento, incluyendo la actualización regular del sistema, el monitoreo continuo y la documentación detallada, aseguran que la aplicación se mantenga operativa y relevante a largo plazo. La capacidad de realizar actualizaciones y ajustes con mínima interrupción es crucial para la sostenibilidad del proyecto.

**Conclusión Final:**

* El proyecto de desarrollo del repositorio de juegos retro ha cumplido con sus objetivos principales de proporcionar una plataforma robusta, segura y rica en funcionalidades. La elección de tecnologías y la atención a los detalles de diseño y seguridad han resultado en una aplicación que no solo satisface las necesidades actuales de los usuarios, sino que también está preparada para futuras expansiones y mejoras. La aplicación no solo actúa como un repositorio informativo, sino que también crea una comunidad interactiva y dinámica para los entusiastas de los juegos retro.