**Documentación**

**(Juego de serpientes en tiempo real)**

**Palomino Aldazabal, Roy Iván**

**Condori Torres, Juan Gabriel**

***Universidad Nacional José María Arguedas***

**ÌNDICE**

[Introducción 3](#_Toc394238639)

[Métodos 4](#_Toc394238640)

[WEBSOCKETS 4](#_Toc394238641)

[COMUNICACIÓN CLIENTE – SERVIDOR 5](#_Toc394238642)

[NODE.JS 5](#_Toc394238643)

[EXPRESS.JS 5](#_Toc394238644)

[¿QUÉ ES JAVASCRIPT? 5](#_Toc394238645)

[DOCUMENTO DE REQUERIMIENTOS DE JUEGO SNAKE EN TIEMPO REAL 6](#_Toc394238646)

[Requerimientos funcionales: 6](#_Toc394238647)

[Requerimientos no funcionales: 7](#_Toc394238648)

[Diagrama de casos de uso: 7](#_Toc394238649)

[Diagrama de diseño: 7](#_Toc394238650)

[Diagrama de Análisis: 8](#_Toc394238651)

[RESUMEN GENERAL DEL JUEGO 8](#_Toc394238652)

[RESUMEN GENERAL DEL CODIGO FUENTE 8](#_Toc394238656)

[Entorno y escenario 9](#_Toc394238665)

[Los actores. 10](#_Toc394238667)

[El movimiento y las interacciones. 12](#_Toc394238668)

[Los controles. 13](#_Toc394238669)

[La conexión. 14](#_Toc394238670)

[Resultados 15](#_Toc394238671)

[Conclusión 15](#_Toc394238672)

# Introducción

Una de las características más apreciadas de jugar en Internet, es el poder jugar con otras personas alrededor del mundo. Para poder programar un juego de esta naturaleza, es necesario conectar a cada uno de los jugadores mediante un servidor, el cual se encarga de manejar los datos de cada jugador y enviarlos a los demás mediante sockets.

Es siguiente proyecto tiene la finalidad de desarrollar un juego dinámico en tiempo real con estrategias y métodos de como jugar. Usando las tecnologías de programación y realizada en una plataforma web, se programara en JavaScript y se convertirá en un juego multiusuario usando las herramientas NODE.js y Socket.io (mediante el cual permitirá de forma rápida y ágil ser una conexión de puertos entre computadores y/o navegadores).

El funcionamiento del juego consistirá en que un jugador utilice las teclas de direccionamiento para encontrar comer la mayor cantidad de comida, la finalización del juego se dará cuando el agente te atrape y destruya, como el juego es multiusuario el que coma más comida es el ganador.

Pero el principal problema se da en la parte de la conexión que por cierto se dará mediante el navegador, usando puertos intentaremos hacer la conexión con las herramientas antes mencionadas.

Finalmente este documento probablemente será mejor entendido por los lectores que tengan conocimiento en lenguajes orientado al objeto, como Ruby, Python, PHP o Java; teniendo poca experiencia con JavaScript, y ninguna experiencia en Node.js. 

# Métodos

En esta parte del documento presentaremos y conoceremos los métodos para desarrollar nuestro juego multiusuario así como, lenguajes de programación, herramientas, diagrama de caso de uso, entre otros.

Herramientas a utilizar:

* Lenguaje de programación (JavaScript)
* Herramienta node.js (interconexión mediante puertos)
* Herramienta node.js (interconexión mediante puertos)
* Herramienta socket.io (interconexión mediante navegadores)

## WEBSOCKETS

HTML5 como incorpora Websockets entre sus nuevas características que es lo que se define como: comunicación entre cliente y servidor.  Entonces

**¿QUE ES WEBSOCKETS?**

Websockets es la tecnología que llega para resolver los problemas de comunicación que plantea el esquema anteriormente descrito. De manera simple, Websockets permite comunicar el cliente y el servidor a través de un canal Full Duplex bidireccional.

## COMUNICACIÓN CLIENTE – SERVIDOR

La problemática que intenta resolver websockets, generalizando, es comunicar los procesos del servidor con los procesos del cliente (navegador) en una plataforma web, mejorando las técnicas actuales.  
Para los que no conocen la plataforma de desarrollo web básica esta se compone generalmente de un lado cliente y un lado servidor, que interactúan mediante peticiones.

## NODE.JS

Es un entorno de programación en la capa del servidor basado en el lenguaje de programación JavaScript, con I/O de datos en una arquitectura orientada a eventos y basado en el motor JavaScript V8. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web.

Al contrario que la mayoría del código JavaScript, no se ejecuta en un navegador, sino en el servidor.



## EXPRESS.JS

Express.js está basado en Connect, que es un framework extensible de manejo de servidores HTTP, el cual provee plugins de alto rendimiento conocidos como middleware. Middleware es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos.

## ¿QUÉ ES JAVASCRIPT?

JavaScript es un sencillo lenguaje de programación, creado y diseñado para utilizarse en las páginas web.  
Permite añadirles muchas funcionalidades, que las hacen más dinámicas, interactivas, útiles y atractivas.  
JavaScript no es más que líneas de texto, que pueden ser escritas con cualquier editor, como el sencillo Bloc de notas de Windows.  
Este código es insertado dentro del código HTML o código fuente de las páginas web, encerrado entre etiquetas que le indican al navegador que son instrucciones que debe ejecutar.

**JQuery**

Es una biblioteca JavaScript la misma que permite la manipular documentos HTML, manejar eventos, interactuar mediante Ajax, personalizar animaciones mediante una API fácil de usar que trabaja en múltiples navegadores. Actualmente es una de las bibliotecas JavaScript más utilizadas en el desarrollo de aplicaciones Web debido a sus numerosos plugins e inclusive su aplicación se la realizar en dispositivos móviles mediante su versión adaptada jQuery Mobile

**Socket.io**

La meta de socket.io es hacer posible aplicaciones en tiempo real en todos los navegadores y dispositivos móviles, evitando a los programadores las diferencias entre éstos y todas sus complicaciones.

socket.io, es un paquete para Node.js y también para Express, podemos crear aplicaciones Ajax en los dos sentidos, cosa que antes, podía ser una labor extremadamente complicada, es decir, tanto de comunicación entre el cliente y el servidor, como viceversa, del servidor al cliente.

Para más información visite: <http://socket.io>

Para poder tener más conocimiento acerca de la instalación y manipulación de las herramientas utilizadas te invitamos a visitar los sitios tutoriales siguientes:

[nodejs.org](http://nodejs.org/)

<http://opengameart.org/node/17167>

<http://librosweb.es>

<http://www.nodebeginner.org/>

# DOCUMENTO DE REQUERIMIENTOS DE JUEGO SNAKE EN TIEMPO REAL

## Requerimientos funcionales:

Realizar un juego en tiempo real usando nuevas tecnologías basadas en las comunidades de usuarios, como navegador Google Chrome.

* 1. Identificar cada jugador con su agente y su posición.
  2. El juego permite usar las teclas de direccionamiento para su desplazamiento de la serpiente.
  3. Permite visualizar a todos los jugadores en tiempo real.
  4. Permite al jugador comer la fruta en su camino.
  5. Permite el ingreso de nuevos jugadores.
  6. Permite que el agente asesine a nuestra serpiente con tan solo una colisión.
  7. El usuario puede jugar contra otro usuario por red.
  8. El usuario puede jugar solo.
  9. El sistema contara con una interfaz gráfica predeterminada, debe ser de fácil uso y entrenamiento por parte de los usuarios
  10. Reconocer la finalización de un juego (atrapado por el agente).

## Requerimientos no funcionales:

1. Permitir visualizarse y funcionar correctamente en cualquier el navegador, Google Chrome versiones posteriores a V8.
2. El juego no debe tardar en cargar más de 4 segundos en mostrar las interfaces y a los jugadores ingresantes.
3. Acondicionar el juego a las diferentes resoluciones de pantalla.
4. No colapsara debido al número de usuarios en línea, soportara (8) usuarios por el momento.

## Diagrama de casos de uso: en nuestro juego solo encontramos 3 casos de uso que son los siguientes:

## Realizado en el programa starUML.

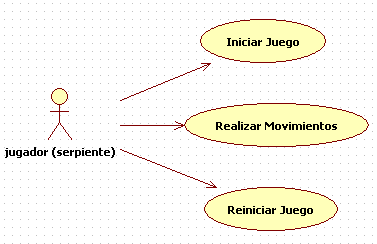


Figura 1: Diagrama de caso de usos.

## Diagrama de diseño:

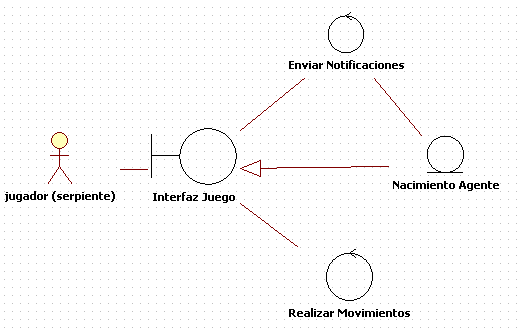


Figura 2: Diagrama de diseño.

## Diagrama de Análisis:

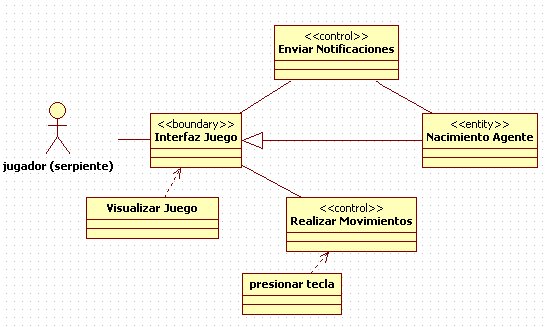


Figura 3: Diagrama de análisis.

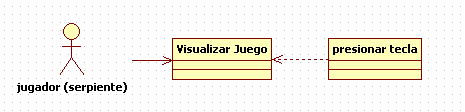


Figura 4: Diagrama de análisis detallado.

**Diagrama de clases:**

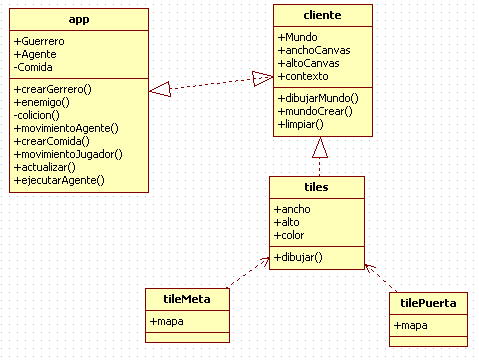
****

Figura 5: Diagrama de clases.

# RESUMEN GENERAL DEL JUEGO

El juego se planteara de la siguiente manera:

Existirá una serpiente (jugador) quien es el bueno del juego que será perseguido por un agente en su afán de matarlo, el juego consiste en que debemos de comer las comidas (frutas) en nuestro camino y se gana siempre en cuando crezcamos lo suficiente, ojo que también se pueden comer entre serpientes, para eliminar a otro jugador que entro en tiempo real.

Cada jugador que nuevo ingresara con su respectivo agente y cada vez que se actualice se cambiara la dirección de la comida.

# RESUMEN GENERAL DEL CODIGO FUENTE

Bueno dicho la lógica del juego el código está dividido en 3 clases:

**Clase (cliente: cliente.js):** en esta clase se detalla todo lo que el cliente solicita al servidor, mediante métodos llamaremos a nuestro jugador como el método caminar, comer, dibujar el mapa de mosaicos.

**Clase (servidor: app.js):** clase más importante que recibirá las peticiones y realizara la creación de nuevas serpientes y agentes.

**Clase (tiles: tiles.js):** en esta clase enviaremos parámetros para que dibuje el mapa mediante la técnica de los tiles o mosaicos, pasándole parámetros para su creación de nuestro escenario llamado mundo.

Pero se preguntaran sonde entras las demás herramientas HTML5 Y CANVAS están en index.html, y haremos la conexión entrelazando las librerías:

***Socket.io***

***Jquery.min***

También introducimos código de **NODE.js** para la comunicación cliente-servidor-

# Entorno y escenario

Bueno como dijimos anteriormente vamos a crear un juego que funcione correctamente en el siguiente navegador: Chrome. El juego será un clásico conocido como “juego de serpientes”. Como contenedor del juego utilizaremos el nuevo elemento de HTML5: *CANVAS.*

El mapa se creara usando mosaicos o también conocidos como tiles los cuales nos permitirán dibujar en mapa en 2d mediante matrices y dibujándolos en un cambas, para tener mayor conocimiento de los tiles se recomienda ver:

<http://opengameart.org/node/17167>

<http://librosweb.es>

[http://sinrega.org](http://sinrega.org/xmlrpc.php)

http://blogtechdencias.azurewebsites.net/blog/2013/03/26/creando-tiles-y-notificaciones-as-easy-as-pie/

Teniendo ese previo conocimiento creamos nuestro mundo en la clase cliente y lo dibujamos con el método *dibujarMapa()* después envía todo el mapa a la clase tiles para poder dibujar posteriormente

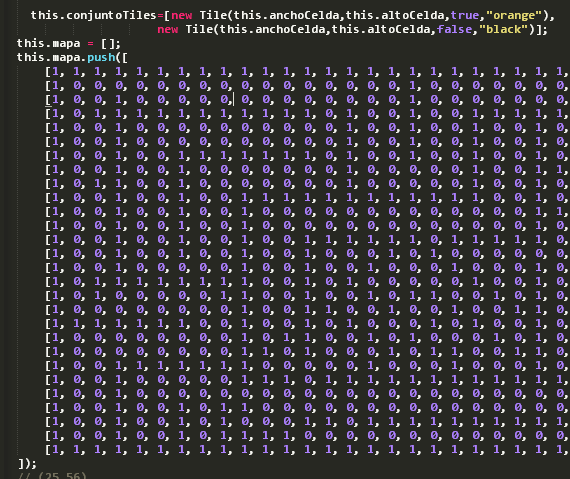


Figura 6: Dibujo del mapa mediante mosaicos.

En esto consiste los mosaicos podemos cambiarlo según nuestro criterio y se dibuja utilizando el método:

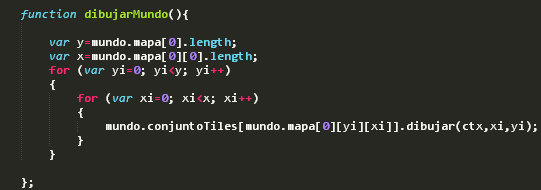


Figura 7: Método dibujar mundo.

# Los actores.

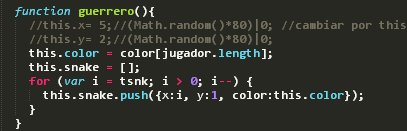
Pues bien vamos a presentar a los actores de nuestro juego. En primer lugar "la serpiente", que será el bueno del juego. Quien tiene que comer la mayor cantidad de frutas

El jugador manejará a la serpiente con las teclas de direccionamiento derecha, izquierda, arriba, abajo para el movimiento.

Como mencionamos en la parte inicial toda esta lógica se encuentra en nuestra clase app.js de quien hablaremos a continuación:



Método para crear al guerrero:



Instancia para crear a los jugadores en este caso el jugador y el enemigo:



Ahora nuestro jugador tendrá su movimiento por el cual nosotros restringimos que no deben pasarse los muros o respetarlos y esto lo conseguimos mediante el siguiente código

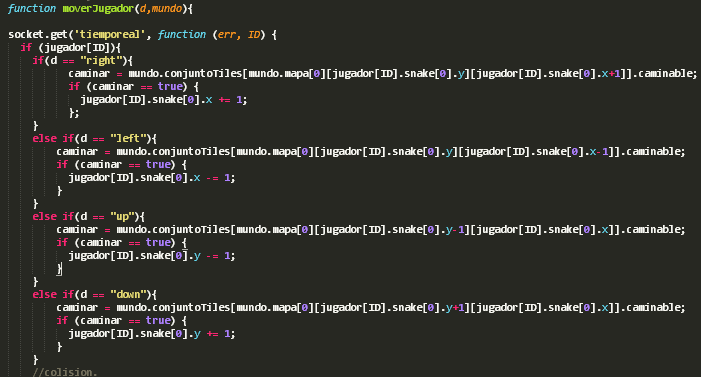


Figura 8: método de mover a la serpiente.

Nos atacará un enemigo llamado agente de quien tenemos que escapar y evitar que nos exterminen. Los enemigos harán la búsqueda de forma inteligente sobrepasando obstáculos, buscando la manera de eliminar a la serpiente y sin respeto a nadie.



Del mismo modo que creamos a nuestro guerrero también creamos a nuestro agente la diferencia es que tiene su método para poder buscar a la serpiente:

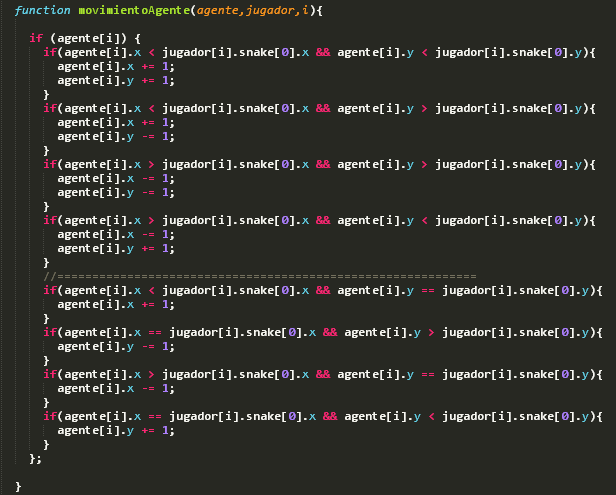


Figura 9: método del movimiento del agente.

# El movimiento y las interacciones.

El movimiento en el CANVAS es muy sencillo. Esto se actualiza en función a unos ciclos de reloj, según el cual, en cada ciclo se actualiza el estado de los bits que maneja. Tenemos una función *actualizar()* que será quien nos marque los pulsos. Después de cada pulso nosotros actualizamos el estado del CANVAS.

Para comprobar las interacciones y actualizar a los actores usamos el método *actualizar*() que comprobará cosas necesarias en cada pulso como son:

* Si entro un nuevo jugador (serpiente).
* Si hemos ganado contra los demás jugadores del juego.
* Si hemos comido una fruta y el lugar donde se encuentra.
* Si el agente nos encontró y nos destruyó.



Figura 10: Actualiza al jugador.

**Los controles**

Como en cualquier juego, el usuario deberá interactuar con el jugador que maneja.

En nuestro caso deberá manejar a la serpiente para esquivar a los agentes y para poder comer las frutas Los controles serán los siguientes:

* Flecha izquierda: mueve la serpiente a la izquierda.
* Flecha derecha: mueve la serpiente a la derecha.
* Flecha arriba: mueve la serpiente a la arriba.
* Flecha abajo: mueve la serpiente a la abajo.

Como vimos en los puntos anteriores, el juego se actualiza en base a unos pulsos. Por lo que no nos preocuparemos por cada movimiento tanto del jugador como el agente son similares solo varia en la lógica de caminar

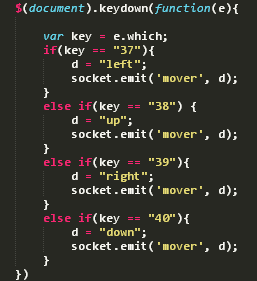


Figura 11: Lógica de movimiento de la serpiente

# La conexión

Esto se llega a lograr mediante las herramientas antes mencionadas y código socket.io como podemos apreciar en la siguiente imagen

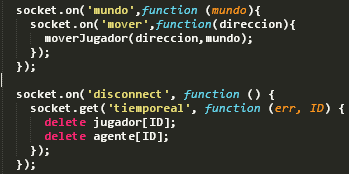


Figura 12: Código de la conexión.

La conexión son peticiones que se reciben en el servidor y enviadas por el mismo al cliente. De hecho hay mucho más código fuente para realizar su conexión por lo mismo le recomendamos leer artículos referentes a los temas de comunicación e implementación de programas en tiempo real en una plataforma web.

# Resultados

Bueno como pudimos observar líneas arriba nuestro juego es muy dinámico y en tiempo real, pero no se cumplió con el objetivo de una interfaz mucho mejor con niveles, puntuaciones, vidas, imágenes y otros.

Pero se cumplió con el objetivo principal que era la conexión en tiempo real entre usuarios.

Además estas herramientas no solo sirven para crear juegos en navegadores u otros sino también para hacer muchas aplicaciones en tiempo real como un chat, pizarras, juegos entre otros.

Es muy interesante y te invitamos a conocerlos ya que es una novedad en lo que se define como programación en plataforma web.

# Conclusión

Llegando a la parte final del trabajo se pudo apreciar que es de gran utilidad hacer programas, juegos, entre otras aplicaciones mediante la web ya que el mundo debe de estar interconectado mediante las llamadas redes.

Como observamos el trabajo se desarrolló de forma rápida y sencilla se cumplió con el principal objetivo que era la interconexión entre navegadores e interactuar con el mismo. Del modo en que solo nos quedaría utilizar el juego realizado anteriormente para poder unirlo y así convertirlo en un juego robusto.

**Agradecimientos**

Agradecemos a todas las personas que nos dieron aliento para continuar con el proyecto y no derrumbarnos en cada problema que se presentó, agradecido por ello y por la comprensión de los padres.

**Referencias**

[1] <http://opengameart.org/node/17167>

[2] <http://librosweb.es>

[3] <https://github.com/joyent/node/wiki/Installation>

[4] [http://socket.io](http://socket.io/)

[5] http://www.project-cyan.com/category/tutoriales-javascript-nivel-intermedio/

**Datos de Contacto:**

*Roy Iván, Palomino Aldazabal.* Institución: UNIVERSIDAD NACIONAL JOSE MARÍA ARGUEDAS.*Dirección talavera.* *E-mail:* [*Royivan\_24@hotmail.com*](mailto:Royivan_24@hotmail.com)

*Juan Gabriel, Condori Torres.*Institución: UNIVERSIDAD NACIONAL JOSE MARÍA ARGUEDAS.*Dirección: Andahuaylas.* *E-mail:* [*juanki\_26\_60@hotmail.com*](mailto:juanki_26_60@hotmail.com)