Universidad Central “Marta Abreu” de Las Villas

Facultad de Matemática, Física y Computación

TRABAJO DE DIPLOMA

Título: Análisis y Diseño de una solución en tiempo real de un juego de ajedrez online.

Autores

Lance Lerone Adrian Chichester

Shivanand Willie

Tutor

Ing. Ernesto Miguel Rodríguez Rodríguez

Santa Clara

2013

Hacemos constar que el presente Trabajo de Diploma ha sido realizado en la facultad de Matemática, Física y Computación de la Universidad Central “Marta Abreu” de Las Villas (UCLV) como parte de la culminación de los estudios de Ingeniería Informática, autorizando a que el mismo sea utilizado por la institución para los fines que estime conveniente, tanto de formato tal como parcial.

Firma del Autor	Firma del Autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y que el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

Firma del Tutor	Jefe del Seminario de Programación

Agradecimientos

Gracias a Dios por darme la fuerza cuando la necesitaba para mantenerme enfocado y comprometido a terminar mis tareas y cumplir mis metas.

Le agradezco mi padre, por su guía, apoyo y consejo durante los momentos difíciles, cuando sentí que no podría terminar exitosamente la carrera, le agradezco por creer en mí cuando yo no tenía fuerzas, y darme la resolución para seguir adelante.

A mi madre, que siempre me demostró su amor sin condiciones, y me sostuvo cuando creí que iba a fallar, le agradezco por enseñarme que en la constancia está el éxito, por darme ánimos para afrontar los momentos tan difíciles que me han tocado vivir lejos de mi patria y de mi familia.

A mi hermana querida, que a pesar de la distancia me tuvo siempre en sus oraciones y en su corazón, y que se alegra tanto de mi éxito.

A mi novia, que me ha apoyado a través de los momentos más difíciles de este último año, dándome fuerzas para cumplir con mis obligaciones, y en momentos de enfermedad y de felicidad ha estado a mi lado.

A todos mis profesores, que me han ayudado durante los seis años que he pasado aquí en Cuba, formándome como ingeniero y como persona, para ellos un reconocimiento muy especial.

A Shivanand, mi compañero en esta tesis, el cual ha sido un amigo entrañable a través de todos estos años.

A mis amigos, que también me apoyaron e hicieron que el tiempo pasado en este país fuera más agradable.

A todas las personas que conozco, las que de una forma u otra me han tendido sus manos, gracias por hacer posible este momento tan especial.

Lance Chichester

Agradecimientos

A la diosa Saraswati Maa para su guía, su conocimiento y su ayuda en los tiempos difíciles.

Al Todopoderoso Namah Shivaya por darme la fuerza, la fe para seguir, para luchar y vencer todos los obstáculos que estaban en mi camino.

Al gobierno de Guyana, Dr. Jagdeo y La Revolución Cubana por otorgarme la oportunidad de ser un Ingeniero Informático.

A mi familia, mis padres, mi abuela, mis tías, mis tíos por siempre estar allí en cualquier momento en que yo necesitaba algo.

A mi tutor Ernesto Miguel por su conocimiento, su paciencia y su motivación en lograr terminar la tesis.

A mis profesores especialmente Carlos Donis, Carlos Villanueve, Deborah y Marie Elena por su ayuda durante los cinco años y con la tesis.

A mi compañero de la tesis Lance, fue un honor trabajar contigo en este proyecto.

A todos mis amigos por su motivación y apoyo para poder terminar esta tesis.

A mi hermano Surendra por su consejo y apoyo en todos mis proyectos.

A mis compañeros de aula por su apoyo, su ayuda durante mi tiempo en ese país. Fue un placer conocerlos a todos ustedes.

Shivanand Willie

Dedicatoria

A mis padres, que han sido y serán siempre el mayor ejemplo de humildad, perseverancia y amor en mi vida. Gracias por estar ahí para mí.

A mi hermano, que murió el 26 de noviembre de 2008, que estoy seguro que estaría orgulloso de mí, y me acompaña siempre en mi corazón.

Lance Chichester

A mi familia por estar allí para mí, por continuar siempre creyendo en mí y por todo su sacrificio para que yo consiguiera mis metas y sueños.

Shivanand Willie

Pensamientos

Liberty is the right of every man to be honest, to think and to speak without hypocracy. José Martí.

Success is to be measured not so much by the position that one has reached in life, as by the obstacles one has overcome trying to succeed. Booker T. Washington.

Resumen

El ajedrez es un juego donde dos jugadores se enfrentan en un tablero de ajedrez de 64 casillas colocadas en forma de matriz de 8 x 8, es uno de los juegos más populares en el mundo entero y jugado por millones de personas en casa, en clubes, en línea, por correspondencia, y en torneos.

Cada jugador empieza el juego con dieciséis piezas: un rey, una reina, dos torres, dos caballeros, dos alfiles, y ocho peones. Cada uno de los movimientos de los seis tipos de piezas es diferente. El objetivo del juego es atacar y capturar las piezas del oponente, con el objeto de colocar bajo una amenaza ineludible su rey, movimiento conocido como 'jaque mate'. Además del jaque mate, el juego puede ganarse por la resignación voluntaria del contrario, que típicamente ocurre cuando el oponente tiene demasiadas piezas perdidas, o si el jaque mate parece inevitable. Un juego también puede resultar en un empate en varias maneras dónde ningún jugador logra la victoria.(FIDE)

Este trabajo se ideó para ofrecerles a los amantes del juego ciencia en la UCLV una solución para lograr que la interacción entre los usuarios sea en tiempo real dándole poco margen de cometer fraude a usuarios tramposos.

Abstract

Chess is a two-player board game played on a chessboard, a square-checkered board with 64 squares arranged in an 8 x 8 grid. It is one of the world's most popular games, played by millions of people worldwide at home, in clubs, online, by correspondence, and in tournaments.

Each player begins the game with sixteen pieces: one king, one queen, two rooks, two knights, two bishops, and eight pawns. Each of the six piece types moves differently. Pieces are used to attack and capture the opponent's pieces, with the object of the game being to 'checkmate' the opponent's king by placing it under an inescapable threat of capture. In addition to checkmate, the game can be won by the voluntary resignation of one's opponent, which typically occurs when too much material is lost, or if checkmate appears unavoidable. A game may also result in a draw in several ways, where neither player wins.

Online Chess is a game for two players who use Black and White chess pieces respectively with the objective of capturing the opponent's king; this is known as checkmate.

This project was made to offer the lovers of the scientific game en UCLV a solution to make the interaction between the users occur in real time and thereby not giving deceitful users an opportunity to cheat.

Índice

Tabla de figuras 1

Introducción 3

Objetivos específicos: 4

Valoración de la investigación 5

Métodos teóricos usados 5

Capítulo I 7

-Marco Teórico de las herramientas usadas en el portal web 7

1.1 Valoración de plataformas para jugar ajedrez en línea. 7

1.1.1 MKGIChessClub-2.2.0 7

1.1.2 Chess.uclv.edu.cu 8

1.1.3 Compwebchess_v2.2 8

1.2 Principales Conceptos 9

1.2.1 Aplicación Web 9

1.3 Principales Herramientas de la tecnología 9

1.3.1 RUP (Proceso Unificado de *Rational*) 9

1.4 Lenguaje Modelado 10

1.4.1 UML (Lenguaje Unificado de Modelado) 10

1.5 Herramienta CASE 10

1.5.1 Visual Paradigm 10

1.6 IDE de Programación 11

1.6.1 Adobe Dreamweaver 11

1.7 Lenguajes de Programación 11

1.7.1 Lado cliente 11

1.7.2 Lado Servidor 12

1.8 Servidor Web 13

1.8.1 XAMPP 13

1.9 Gestor de Base de Datos 14

1.9.1 MySQL 14

Capítulo 2 16

-Análisis y diseño del sistema 16

2.1 Introducción 16

2.2 Objeto de estudio 16

2.2.1 Problema y situación problemática 16

2.3 Especificación de los requisitos del software 16

2.3.1 Requisitos funcionales 17

2.3.2 Requisitos no funcionales 17

2.4 Objeto de automatización 18

2.4.1 Socket.io 18

2.4.2 Node.js 19

2.4.3 Comet 21

2.4.4 Pushlet 26

2.4.5 JQuery 32

2.5 Conclusiones parciales 34

Capítulo III 36

- Implementación y prueba 36

3.1 Introducción 36

3.2 Sistema de registro y autenticación 36

3.3 Descripción de Node.js 37

3.4 Configuración de Los Ficheros de JavaScript 37

3.5 Configuración de las Carpetas 43

3.6 Configuración de los módulos 44

3.6.1 Forever 45

3.6.2 Request 47

3.6.3 MySQL 48

3.6.4 Socket.io 50

3.7 Arrancando el sitio 55

3.8 Pruebas del sitio _____ **59**

Conclusiones _____ ***61***

Recomendaciones _____ ***63***

Bibliografía _____ ***64***

Tabla de figuras

figura 3 1 38
figura 3 2 38
figura 3 3 39
figura 3 4 39
figura 3 5 40
figura 3 6 40
figura 3 7 41
figura 3 8 41
figura 3 9 42
figura 3 10 43
figura 3 11 44
figura 3 12 45
figura 3 13 45
figura 3 14 46
figura 3 15 46
figura 3 16 47
figura 3 17 48
figura 3 18 49
figura 3 19 49
figura 3 20 50
figura 3 21 51
figura 3 22 52
figura 3 23 52
figura 3 24 53
figura 3 25 53
figura 3 26 54
figura 3 27 54
figura 3 28 55
figura 3 29 56
figura 3 30 56
figura 3 31 57
figura 3 32 58

figura 3 33 _____ *59*
figura 3 34 _____ *59*
figura 3 35 _____ *60*
figura 3 36 _____ *60*

Introducción

"El ajedrez es el gimnasio de la mente" – Blaise Pascal (ChessQuotes.com, 2012)

"Cuando usted está solo, cuando usted se siente un forastero en el mundo, juega Ajedrez. Esto levantará sus espíritus y será su consejero en la guerra". – Aristóteles (ChessQuotes.com, 2012)

El origen del ajedrez, llamado "Chataranga" puede ser remontado al siglo VI en la India, durante el Imperio de Gupta, donde sus piezas representaron cuatro divisiones militares - la infantería, caballería, elefantes y coches que han evolucionado respectivamente en los días modernos al peón, caballero, alfil y torre (Josten, (2001)).

El juego llegó a Europa occidental en el siglo IX, donde continuó evolucionando. Alrededor de 1475, varios cambios significativos hicieron del juego lo que es hoy. Los peones ganaron la opción de adelantar dos cuadrados en su primer movimiento, mientras alfiles y reinas adquirieron sus habilidades modernas. La reina reemplazó el visir en el fin del siglo X y por el siglo XV se convirtió en la pieza más poderosa dentro del juego; por eso el ajedrez moderno es llamado "Ajedrez de la Reina" o "Queen's Chess". Estas nuevas reglas rápidamente se extendieron a lo largo de Europa occidental; para distinguirla de sus predecesoras, esta versión es a veces llamada "ajedrez occidental" o "ajedrez internacional" (Calvo).

Durante el XIX siglo, el juego se populariza rápidamente, al aparecer muchos clubes de ajedrez y secciones de ajedrez en los periódicos. Los primeros torneos de ajedrez comenzaron en el fin del siglo XIX con campeonatos acontecidos en diferentes países de Europa (Adams, 2006).

El juego ha evolucionado durante los siglos desde que fue descubierto. El ajedrez moderno sólo se parece al ajedrez antiguo en las piezas que se usan. La forma de jugar ha cambiado radicalmente con reglas diferentes que se adoptan según pasa el tiempo. La tecnología también ha revolucionado el juego, las personas en las diferentes partes del mundo pueden disfrutar "el juego anciano" simplemente usando computadores y una conexión a Internet.

El ajedrez contemporáneo es un deporte organizado con ligas internacionales y nacionales estructuradas, torneos y congresos. La organización internacional de ajedrez es conocida como FIDE (por sus siglas en francés "*el Fédération el des de Internationale Échecs*").

El sitio "chess.uclv.edu.cu" fue diseñado para personas que desean jugar ajedrez. El sitio provee un interfaz gráfico para facilitar la interacción entre los jugadores, no obstante, la interacción es insuficiente y se le podrían incorporar mejoras que serían bien vistas por los usuarios que lo visitan.

El sistema del juego tiene defectos, los juegos demoran muchísimo porque no hay tiempo límite, hay personas que toman un día para mover una pieza, lo cual no debe permitirse, ya que esa persona tendría tiempo suficiente para consultar con otra(s) persona(s) o software inteligente de ajedrez para saber cuál pieza puede mover. Una solución sería jugar en tiempo real, de esta forma evitaríamos en gran medida el fraude que se comete en el sitio actual que tiene la UCLV.

Teniendo en cuenta lo anteriormente planteado, se formula el siguiente **problema a resolver:**

¿Cómo desarrollar una solución que permita a los usuarios de la UCLV jugar ajedrez en tiempo real?

A partir del problema planteado se define como **objeto de estudio**: los sistemas de juegos ajedrez online en tiempo real.

Debido a la necesidad de darle solución a la problemática planteada se decidió cumplir con el siguiente **objetivo general:**

Desarrollar una solución que permita incorporar al sitio "chess.uclv.edu.cu" la interacción en tiempo real de sus usuarios.

Objetivos específicos:

1. Valorar las principales plataformas web de ajedrez online.

2. Valorar de las principales tecnologías en el campo de la programación Web así como la selección y asimilación de las necesarias para la implementación de la solución.
3. Realizar un estudio sobre los sistemas en tiempo real y su modelamiento.
4. Definir patrones de arquitectura, diseño y implementación.
5. Analizar y estudiar la aplicación "chess.uclv.edu.cu" y los servicios que esta brinda.
6. Realizar una entrevista a maestros de ajedrez sobre la importancia y utilidad de la aplicación.

Valoración de la investigación

A partir de una entrevista con el Gran Maestro de ajedrez Jesús Nogueira, que ocupa el lugar 125 entre los mejores jugadores a nivel mundial, se obtuvieron ideas sobre las características que debe tener el sitio de ajedrez, de acuerdo a las necesidades y el nivel de los jugadores.

Según acoto el Gran Maestro, el sitio debe poseer una interfaz sencilla, comprensible para los jugadores menos experimentados, y diferentes opciones de tiempo para cada jugada. Además sería necesario contar con un sistema de "Elo" que permita establecer cuáles son los mejores jugadores.

El Maestro estuvo de acuerdo con la idea de la investigación, ya que en su opinión puede ser una excelente oportunidad para que jugadores de toda Cuba intercambien experiencias y disfruten de juegos con calidad, y puede contribuir a elevar el nivel de juego de los ajedrecistas de todo el país.

Métodos teóricos usados

- **Analítico-sintético:** en esta investigación se emplea este método para analizar las teorías, documentos y todo tipo de información que se tiene sobre el desarrollo del software; permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio y

lograr así una mejor comprensión del mismo. También se utiliza para el análisis de los resultados más significativos que se han obtenido actualmente en ese campo.

- **Histórico-lógico:** a través de este método se logró un mayor entendimiento y conocimiento de las diferentes técnicas y herramientas de modelado a utilizar, dando la posibilidad de analizar la trayectoria histórica de los mismos, así como su evolución y desarrollo.
- **Modelación:** este método es de gran importancia para la investigación debido a que brinda la posibilidad de representar las propiedades y funcionalidades que tendrá el sistema a desarrollar**.**

Como **métodos empíricos** se utilizó:

- **Observación:** con este método se logró recopilar información acerca de los principales conceptos y características de los diferentes analizadores estáticos.

El documento está estructurado en 3 capítulos:

- **Capítulo 1 :** Marco Teórico de las herramientas usadas en el portal web de ajedrez
- **Capítulo 2 :** Análisis y diseño del sistema
- **Capítulo 3 :** Implementación y prueba

Capítulo I

-Marco Teórico de las herramientas usadas en el portal web

1.1 Valoración de plataformas para jugar ajedrez en línea.

1.1.1 MKGIChessClub-2.2.0

Es una aplicación libremente transferible del Internet, la cual posee las funciones básicas de un portal web para jugar ajedrez. Está basado en el lenguaje PHP.

Ventajas:

- El código fuente es sencillo y fácil de entender.
- Posee una interface fácil de usar.
- Posee un sistema de clasificación basado en ELO que proporciona una buena representación de las habilidades del jugador.

Desventajas:

- El diseño de la interface no tiene atractivo.
- No posibilita una buena comunicación entre los jugadores, ya que no está disponible ningún chat u otro medio para transmitir información entre jugadores.
- No ofrece opciones de tiempo para el juego.
- El juego no está en tiempo real, o sea, cuando un jugador mueve sus piezas, la jugada no se actualiza automáticamente en la pantalla del otro jugador. Esto posibilita que los jugadores hagan trampas, apoyándose en programas u otros tipos de ayudas para ganar.
- No ofrece muchas opciones de juego, por ejemplo campeonatos, clubes, torneos etcétera.

1.1.2 Chess.uclv.edu.cu

Es el sitio que usa la Universidad Central de Las Villas (UCLV) para brindar el juego de ajedrez, el cual presenta deficiencias en la estructura y funcionalidad. Esencialmente tiene las mismas ventajas y desventajas que el anterior, ya que está basado en el mismo *framework*, sin introducir cambios significativos.

1.1.3 Compwebchess_v2.2

Esta es una aplicación web para jugar ajedrez, que ofrece al usuario muchas opciones de juego, así como la oportunidad de personalizar su perfil. Está hecho en PHP y Java Script.

Ventajas:

- El jugador tiene muchas opciones de juego, como torneos, campeonatos o enfrentamiento con un solo contrincante.
- Están disponibles varias opciones de tiempo, mínimo un día y máximo catorce días por juego.
- Tiene un buen sistema estadístico.
- Posee una excelente presentación en los perfiles de los usuarios: se pueden subir fotos y ver toda la información del perfil del usuario de manera integral.
- Tiene un buen sistema de chat que permite a los jugadores comunicarse mientras están jugando.
- Se puede cambiar el idioma del sitio según las necesidades del usuario.

Desventajas:

- La interface del sitio web carece de atractivo visual.
- No hay forma de jugar un juego rápido, por ejemplo, en media hora.
- El juego no transcurre en tiempo real, no se actualiza automáticamente.

Se determina basado en sus ventajas y desventajas que Compwebchess_v2.2 es la mejor de las plataformas investigadas y una adaptación de la misma pueden ser usados en el sitio de ajedrez en la UCLV mejorando significativamente el servicio proporcionado.

1.2 Principales Conceptos

1.2.1 Aplicación Web

Una aplicación web es aquella aplicación que el usuario puede utilizar accediendo a un servidor web a través de internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.(Wikipedia, 2012a)

Con el uso de una aplicación web se puede mejorar la manera de jugar un juego clásico como el ajedrez.

1.3 Principales Herramientas de la tecnología

Metodología de desarrollo de software

Es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.(Wikipedia, 2011)

1.3.1 RUP (Proceso Unificado de *Rational*)

El Proceso Unificado de Rational (*Rational Unified Process* en inglés, habitualmente resumido como **RUP**) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

(Beck, 1999).

¿Por qué se usa?

La simplicidad del método y la comunicación son extraordinariamente complementarias. Es el método que los programadores han estudiado y en el que tienen más dominio. Este método ilustra mejor los procesos del sistema y el diseño.

1.4 Lenguaje Modelado

El lenguaje de modelado es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de software

1.4.1 UML (Lenguaje Unificado de Modelado)

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables (Perdita Stevens, 2002).

1.5 Herramienta CASE

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero.(Wikipedia, 2012b)

1.5.1 Visual Paradigm

Es un software usado para modelar diferentes aspectos de una aplicación o modelo de software. También provee generación de reportes y capacidades de ingeniería de códigos. Además puede generar diagramas a partir de códigos y proporciona ingeniería de varios lenguajes de programación.

¿Por qué se usa?

Se utiliza esta herramienta porque es la que más facilidades ofrece para los autores, y con la cual tienen un mayor nivel de familiarización y dominio.

1.6 IDE de Programación

Un entorno de desarrollo integrado, llamado también **IDE** (sigla en inglés de *integrated development environment*), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.(Wikipedia, 2013)

1.6.1 Adobe Dreamweaver

Es una aplicación en forma de suite que esta destinada a la construcción, diseño y edición de sitio, videos y aplicaciones Web basadas en estándares. Dreamweaver es el programa mas utilizado en el sector del diseño y la programación web por sus funcionalidades y su integración con otras herramientas.

¿Por qué se usa?

La gran ventaja de este editor sobre otros es su gran poder de ampliación y personalización. Se puede programar y editar el contenido web con facilidad y eficiencia. Además es el programa que hemos usado más a menudo en nuestra carrera y es el que tenemos mayor habilidad.

1.7 Lenguajes de Programación

1.7.1 Lado cliente

HTML (Hyper Text Markup Language)

Es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.(Mora, 2001)

CSS

CSS u hojas de estilo en cascada (*Cascading Style Sheets*) es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML. *Cascading* se refiere al orden en que los estilos diferentes son aplicados. CSS es una alternativa popular para codificar los elementos de estilo en páginas Web.(Shelly, 2009)

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario (Pérez, 2008).

JavaScript es discutiblemente el lenguaje de programación ampliamente usada en el mundo (Gosselin, 2011).

AJAX

Ajax, acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones (ERIC PASCARELLO DAVE CRANE, 2006).

1.7.2 Lado Servidor

PHP (*Hypertext Pre-processor*)

PHP (*Hypertext Pre-processor*) es un lenguaje de secuencia de comandos de servidor diseñado específicamente para la Web. Dentro de una página Web se puede incrustar un

código PHP que se ejecutará cada vez que se visite una página. El código PHP es interpretado en el servidor Web y genera un código HTML (Luke Welling).

El lenguaje PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores, el número de sitios en PHP ha compartido algo de su preponderante dominio con otros nuevos lenguajes no tan poderosos desde agosto de 2005. El sitio web de Wikipedia está desarrollado en PHP. Es también el módulo Apache más popular entre las computadoras que utilizan Apache como servidor web.

¿Por qué se usa?

Se usa el Lenguaje PHP porque presenta las siguientes características:

- Gratuito. A1 tratarse de un software libre, puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera completamente libre.
- Enorme eficiencia. Con escaso mantenimiento y un servidor gratuito, puede soportar sin problemas millones de visitas diarias.
- Sencilla integración con múltiples bases de datos (en nuestro caso MySQL). Esencial para una página Web verdaderamente dinámica, es una correcta integración con base de datos.

Además, es el lenguaje que se estudia en la carrera de Ingeniería Informática y del que los autores del presente trabajo tienen mayor conocimiento.

1.8 Servidor Web

Un servidor web es un programa informático que procesa una aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente (Just van den Broecke, 2006).

1.8.1 XAMPP

XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl.

El programa actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas.

Oficialmente, los diseñadores de XAMPP sólo pretendían su uso como una herramienta de desarrollo, para permitir a los diseñadores de sitios webs y programadores testear su trabajo en sus propios ordenadores sin ningún acceso a Internet. En la práctica, sin embargo, XAMPP es utilizado actualmente como servidor de sitios Web, ya que, con algunas modificaciones, es generalmente lo suficientemente seguro para serlo.

1.9 Gestor de Base de Datos

El gestor de base de datos decide cómo se guardan, se ordenan y se recuperan los datos, y también controla el acceso del usuario. El usuario no puede acceder a los archivos de datos directamente, sólo puede comunicarse con el DBMS. El gestor es una barrera que controla el acceso a los datos subyacentes (NORMAN, 2004).

1.9.1 MySQL

El gestor de Base de Datos MySQL usa una arquitectura cliente/servidor que se centra alrededor del servidor.

El servidor de MySQL controla el acceso a los datos para garantizar el uso simultáneo de varios usuarios, para proporcionar acceso a dichos datos y para asegurarse de que solo obtienen acceso a ellos los usuarios con autorización. Por lo tanto, MySQL es un servidor multiusuario y de subprocesamiento múltiple. Utiliza SQL (del ingles *Structured Query Language*, Lenguaje de consulta estructurado), el lenguaje estándar para la consulta de bases de datos utilizado en todo el mundo(Luke Welling).

¿Por qué se usa?

MySQL tiene una gran adaptabilidad a varias plataformas, en nuestro caso como el enlace a la aplicación web. Es fácil de dominar y tiene todas las herramientas de un poderoso gestor de base de datos. También, MySQL es una base de datos muy rápida en la lectura, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

Capítulo 2

-Análisis y diseño del sistema

2.1 Introducción

En este capítulo se exploran las diferentes herramientas para jugar ajedrez en tiempo real y sus características, con la finalidad de encontrar cual es la mejor herramienta utilizable en la implementación del tiempo real en el sitio de chess.uclv.edu.

2.2 Objeto de estudio

2.2.1 Problema y situación problemática

En la Universidad Central de las Villas (UCLV) existe un sitio web donde los usuarios comparten un juego de ajedrez en línea con otros jugadores, utilizando la conexión de intranet universitaria. La mayor deficiencia que existe en el sitio actual es que los jugadores no pueden jugar en tiempo real, o sea, dependen de un servicio de correo que avisa el otro jugador cuando uno ha jugado. Se propone una corrección de la misma, así como un enriquecimiento de la funcionalidad del sitio.

2.3 Especificación de los requisitos del software

El proceso de captura de requisitos tiene gran importancia en el proceso de desarrollo del software, ya que a través de estos se identifica lo que el usuario desea, para finalmente obtener un producto de calidad.

Los requisitos se pueden clasificar en:

- Funcionales.
- No Funcionales.

2.3.1 Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Especifican comportamientos particulares de un sistema. A continuación se presenta el listado de los requisitos funcionales en el sistema que se modela:

- El sistema debe permitir que la interacción con los usuarios sea en tiempo real.
- La tecnología que se utilice debe ser compatible con la plataforma de juego usada.
- Los juegos deben ser de 1, 2 y 3 minutos respectivamente.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales son las características o cualidades que contiene el sistema. Está compuesto de los requisitos de software y requisitos de hardware.

Requisitos de software

- La instalación de un sistema operativo Linux versión 10 o superior o Windows XP o superior.
- Un servidor web como XAMPP, Uniform Server o Wos.
- Un servidor de base de datos MySQL versión 5 o superior.
- La instalación de un navegador Mozilla Firefox versión 6 o superior, Internet Explorer 7 o superior o Opera 10 o superior.

Requisitos de hardware

- Se necesitan 512 MB de memoria RAM, 1 GB de espacio libre en el disco duro y un procesador de 1GHz como mínimo.

2.4 Objeto de automatización

Jugando ajedrez en tiempo real

El juego de ajedrez usualmente requiere comunicación con un servidor y también con otros jugadores para poder funcionar en línea. Este es el fundamento de una experiencia de juego con varios participantes y ha existido con HTTP, donde se usa *POST* y *GET* para manejar juegos. El problema con estos métodos es el retraso, ya que al hacer un cambio la respuesta es demasiado lenta. Los juegos en tiempo real requieren mensajes mandados y recibidos a veces 33-66 veces por segundo, lo cual no es posible usando HTTP solamente. En navegadores modernos se puede disponer de una conexión entre un servidor y un cliente. Según esta conexión se elaboran las diferentes tecnologías que pueden ser usadas.

2.4.1 Socket.io

Socket.io es un poderoso y flexible componente del lado cliente que permite la conexión en tiempo real en el navegador. Este componente apoya tecnologías nuevas como *web sockets*, aún en un fichero HTML capa de transporte y JSON. Más útil sobre todo es su simplicidad y naturaleza que es inherentemente asíncronos, lo cual es extremadamente valioso para escribir códigos en el lado del servidor y el lado cliente.(Rauch, 2011)

Otro beneficio de usar socket.io es el hecho que se adjunta fácilmente con Node.js. Puede servir el lado cliente incluyendo ficheros de juego y datos haciendo que la integración sea limpia y fácil. Cuando se hace la primera conexión, la comunicación con el cliente genera el código JSON automáticamente. Este método funciona con todo los navegadores, incluso de móviles.

Cómo usar el socket.io

En este ejemplo se guardan mensajes en un sitio de chat en tiempo real.

Paso 1: `npm install socket.io ("La instalación")`

Paso 2: `var io = require('socket.io');` ("Llamar el fichero")

Paso 3:

Lado Servidor

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.on('set nickname', function (name) {
    socket.set('nickname', name, function () { socket.emit('ready');
});
  });

  socket.on('msg', function () {
    socket.get('nickname', function (err, name) {
      console.log('Chat message by ', name);
    });
  });
});
```

Paso 4:

Lado Cliente

```
<script>
  var socket = io.connect('http://localhost');

  socket.on('connect', function () {
    socket.emit('set nickname', prompt('What is your nickname?'));
    socket.on('ready', function () {
      console.log('Connected !');
      socket.emit('msg', prompt('What is your message?'));
    });
  });
</script>
```

2.4.2 Node.js

Node.js es un entorno de programación en la capa del servidor basado en el lenguaje de programación *JavaScript*, con una arquitectura orientada a eventos, y basado en el motor *JavaScript*. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web.(Bergström, 2012)

Realmente Node.js es solamente otro contexto que permite a correr código JavaScript en el lado del servidor fuera de un navegador.

Para ejecutar el JavaScript en el servidor, el código necesita ser bien interpretado y ejecutado. Node.js utiliza el ambiente *runtime* V8 VM de *Google* para JavaScript, el mismo usado por el navegador *Google Chrome*.

Incluido en el Node.js existen módulos útiles para diferentes funciones, para que el programador no tenga que escribir el código desde cero. Así, Node.js es un ambiente runtime y también una biblioteca.

¿Cómo se implementa?

Los elementos necesarios para implementar una aplicación con Node.js son:

- Servidor HTTP (es necesario tener un servidor HTTP para correr paginas web)
- Router (el servidor tendría que responder a diferentes peticiones, depende de cual URL pide la solicitud, entonces necesita algún tipo de router para mapear peticiones a los manejadores de peticiones).
- Manejadores de peticiones (para cumplir las peticiones que viene por ruteo, necesita manejadores de peticiones).
- Vista lógica (no solamente se manejan peticiones para los URL, también hay que mostrar el contenido cuando se piden estos URL, lo cual significa que alguna vista lógica puede usar los manejadores de peticiones para mandar contenido al navegador).

Node.js no solamente implementa la aplicación, también el servidor HTTP. De hecho, la aplicación web y su servidor son básicamente iguales.

El siguiente es un ejemplo sencillo de mostrar en la pantalla texto básico.

Servidor básico HTTP (*server.js*)

```
var http = require("http");

function start() {
```

```
http.createServer(function(request, response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.write("Texto aqui");
  response.end();
}).listen(8888);
console.log("Server has started.");
}

exports.start = start;
```

Se ejecuta el código con

```
node server.js
```

Luego, se abre el navegador y se debe puntear en la dirección http://localhost:8888/
Esta muestra una indicación en la pagina web que dice “Texto aquí”.

Ahora puede crear el fichero del lado cliente con la llamada al fichero servidor.

Fichero principal *(index.js)*

```
var server = require("./server");

server.start();
```

Se ejecuta con

```
node index.js
```

Este mostrará el mismo resultado.

Node.js está clasificado número uno entre los entornos de programación construidos para programar aplicaciones en tiempo real, y es usado por cooperaciones tan influyentes como Microsoft, EBay y Yahoo.

2.4.3 Comet

En el desarrollo web, Comet es un neologismo para describir un modelo de aplicación web en el que una petición HTTP mantenida abierta permite a un servidor web enviar datos a un navegador por Tecnología *Push*, sin que el navegador los solicite explícitamente. Comet es un término paraguas de múltiples técnicas para conseguir esta

interacción. Todos estos métodos confían en características incluidas por defecto en navegadores, como JavaScript, en lugar de sobre *plugins* no disponibles por defecto.(Foundation, 2013)

En teoría, el enfoque Comet difiere del modelo original de la web, en el que un navegador solicita una página web completa o trozos de datos para actualizar una página web. En cualquier caso, en la práctica, las aplicaciones Comet usan tradicionalmente programación AJAX con una petición prolongada para detectar nueva información en el servidor. El concepto precede al acuña miento del neologismo, y es conocido por otros nombres, incluyendo *Ajax Push*, *Reverse Ajax*, *Two-way-web*, *HTTP Streaming* and *HTTP server push* entre otros.(PHIL MACARTHY DAVE CRANE, 2008)

La diferencia que existe entre AJAX y Comet es que en este modelo se mantiene una conexión abierta entre el cliente y el servidor web; el cliente no solicita los datos, pero si envía información al servidor, y el servidor no le responde con un bloque de datos, se espera a que haya algún evento de lado del servidor para enviar la información. (PHIL MACARTHY DAVE CRANE, 2008) .

Entre los sitios que usan Comet se encuentran: GMail, Meebo y Facebook.

Volviendo al ejemplo de la aplicación de mensajes instantáneos, el cliente no deberá estar solicitando los cambios de la lista de usuarios, el servidor web es el que le envía los datos al cliente cuando cambia la lista de usuarios, esto reduce considerablemente el consumo de ancho de banda.

- Ejemplo de un sitio que muestra la hora en tiempo real usando un *iframe* escondido.

Los ficheros que necesita son:

- Un fichero PHP que maneja peticiones HTTP (backend.php)

- Un fichero HTML que carga el código JavaScript y que muestra los dato que viene del servidor (index.html)
- Una biblioteca que ayuda escribir el código JavaScript

El la figura 2.1 se muestra como funciona el Comet en esta aplicación:

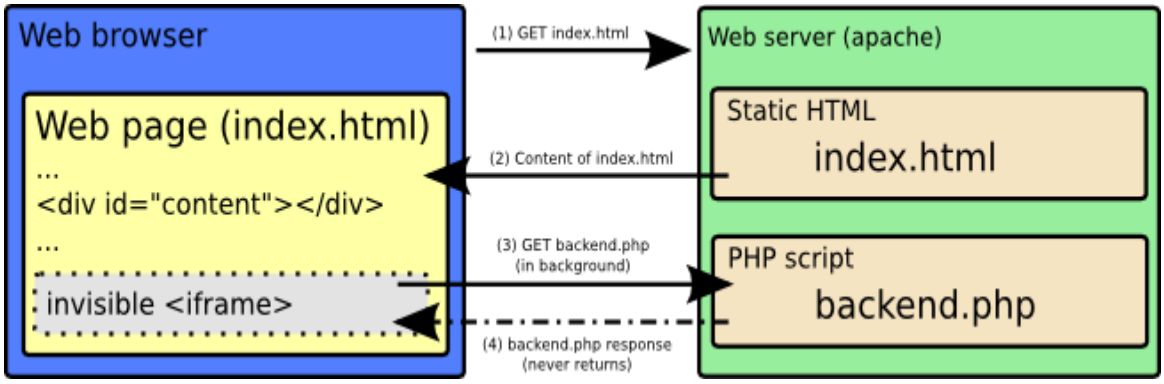

Figura 2 1

El fichero backend.php crea un ciclo infinito que siempre devuelve la hora del servidor mientras que el cliente esta conectado.

```
<?php

  header("Cache-Control: no-cache, must-revalidate");
  header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
  flush();

  ?><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
  <html xmlns="http://www.w3.org/1999/xhtml">
  <head>

    <title>Comet php backend</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
  </head>
  <body>

  <script type="text/javascript">
    // KHTML browser don't share javascripts between iframes
    var is_khtml = navigator.appName.match("Konqueror") ||
navigator.appVersion.match("KHTML");
    if (is_khtml)
    {
      var prototypejs = document.createElement('script');
```

```
      prototypejs.setAttribute('type','text/javascript');
      prototypejs.setAttribute('src','prototype.js');
      var head = document.getElementsByTagName('head');
      head[0].appendChild(prototypejs);
    }
    // load the comet object
    var comet = window.parent.comet;

  </script>

  <?php

  while(1) {
    echo '<script type="text/javascript">';
    echo 'comet.printServerTime('.time().');';
    echo '</script>';
    flush(); // used to send the echoed data to the client
    sleep(1); // a little break to unload the server CPU
  }

  ?>

  </body>
  </html>
```

El documento HTML primera carga la biblioteca prototipo en la etiqueta “<head>”, entonces se crea el reloj del servidor en la etiqueta “ y finalmente crea un objeto JavaScript de Comet que conecta el fichero *backend* al etiqueta “time container”.

index.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
      <title>Comet demo</title>

      <meta http-equiv="Content-Type" content="text/html; charset=utf-
8" />
      <script type="text/javascript" src="prototype.js"></script>
    </head>
    <body>

  <div id="content">
  </div>

  <p>
    <form action="" method="get"
onsubmit="comet.doRequest($('word').value);$('word').value='';return
false;">
      <input type="text" name="word" id="word" value="" />
```

```
      <input type="submit" name="submit" value="Send" />
    </form>
  </p>

  <script type="text/javascript">
  var Comet = Class.create();
  Comet.prototype = {

    timestamp: 0,
    url: './backend.php',
    noerror: true,

    initialize: function() { },

    connect: function()
    {
      this.ajax = new Ajax.Request(this.url, {
        method: 'get',
        parameters: { 'timestamp' : this.timestamp },
        onSuccess: function(transport) {
          // handle the server response
          var response = transport.responseText.evalJSON();
          this.comet.timestamp = response['timestamp'];
          this.comet.handleResponse(response);
          this.comet.noerror = true;
        },
        onComplete: function(transport) {
          // send a new ajax request when this request is finished
          if (!this.comet.noerror)
            // if a connection problem occurs, try to reconnect each 5
seconds
            setTimeout(function(){ comet.connect() }, 5000);
          else
            this.comet.connect();
          this.comet.noerror = false;
        }
      });
      this.ajax.comet = this;
    },

    disconnect: function()
    {
    },

    handleResponse: function(response)
    {
      $('content').innerHTML += '<div>' + response['msg'] + '</div>';
    },

    doRequest: function(request)
    {
      new Ajax.Request(this.url, {
        method: 'get',
        parameters: { 'msg' : request
      });
    }
  }
```

```
var comet = new Comet();
comet.connect();
</script>

</body>
</html>
```

2.4.4 Pushlet

Pushlet es un *framework* que permite el intercambio de mensajes. Una parte de la aplicación se encarga de la generación y envío de mensajes (llamados eventos) al *framework*, la otra parte (usualmente el navegador) recibirá los mensajes. El Nuevo protocolo (V2) también permite a los clientes publicar y subscribirse a eventos. Porque la interfaz puede ser hecha por HTTP (XML) y se puede desarrollar una aplicación en cualquier idioma de programación que se apoye en HTTP. La figura muestra la funcionalidad de Pushlet y como los datos son manejado.(Just van Broecke, 2002)

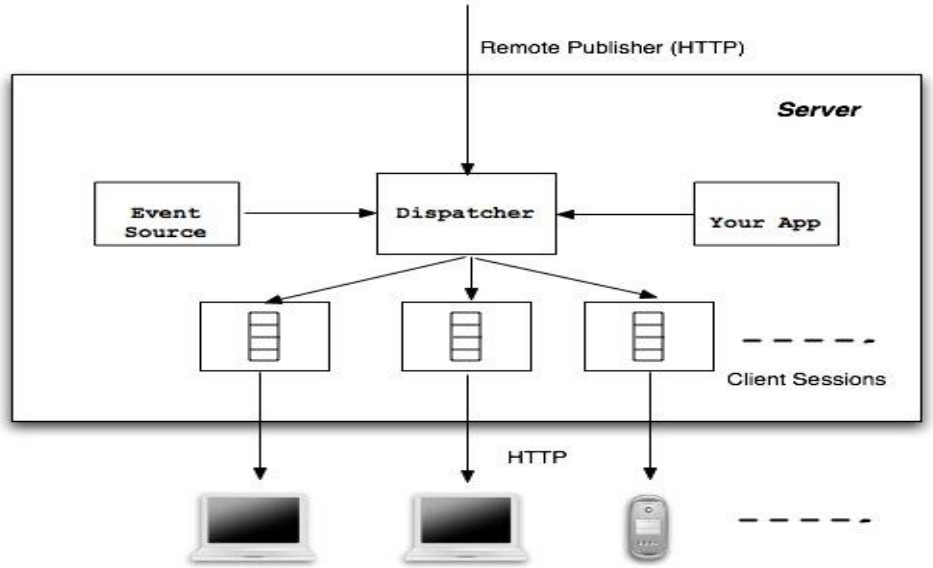

Figura 2 2 (Just van den Broecke, 2006)

Pushlet tiene tres modos de escuchar datos:

- *Stream*: para correr videos
- *Pull*: el cliente inicia la petición de datos del servidor
- *Push*: el servidor hace la petición

Modo de *Stream*

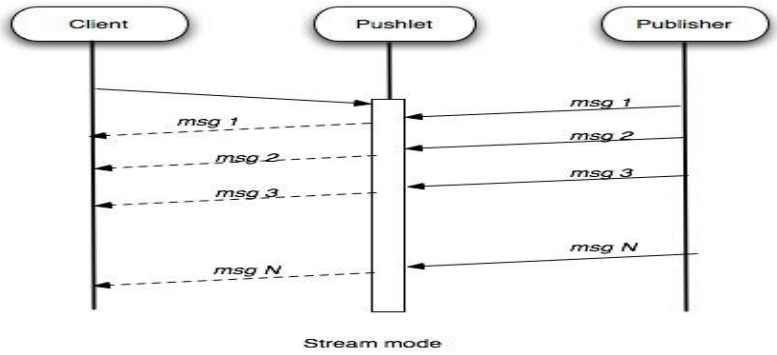

Figura 2 3

Este ejemplo toma la idea de *streaming* que continuamente manda contenido nuevo de HTML al cliente en un ciclo.

```
<HTML>
<HEAD>
   <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-
8859-1">
   <META HTTP-EQUIV="Pragma" CONTENT="no-cache">
</HEAD>
<BODY BGCOLOR="blue" TEXT="white">
<%
  int i = 1;

  try {
    while (true) {
       out.print("<h1>"+(i++)+"</h1>");
       out.flush();

       try {
            Thread.sleep(3000);
```

```
        } catch (InterruptedException e) {
        out.print("<h1>"+e+"</h1>");
         }
      }
    } catch (Exception e) {
        out.print("<h1>"+e+"</h1>");
    }
%>
</BODY>
</HTML>
```

Modo de Pull *(1)* – *Mensaje Bloqueado*

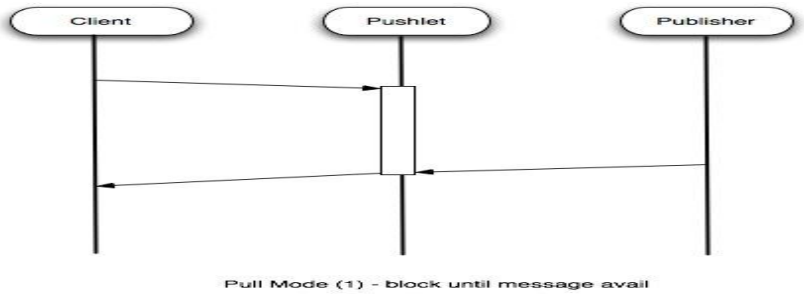

Figura 2 4

Modo de Pull (2) – Mensaje Disponible

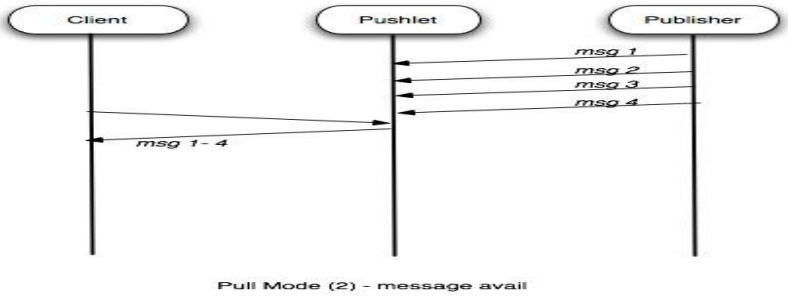

Figura 2 5

El diseño del framework

El *framework pushlet* permite a clientes subscribir a sujetos dentro del servidor desde el cual se reciben los eventos. El diseño básico del framework es Publicar-Subscribir también conocido como Observador y tiene componentes de servidor y de cliente:

- Una colección de clases Java en el lado del servidor diseñado sobre la clase Pushlet.
- Una biblioteca JavaScript reutilizable para el lado cliente (pushlet.js) y de HTML (pushlet.html) para recibir eventos para los clientes DHTML.
- Clases de Java para el lado cliente.
- Generación de eventos de prueba.

La figura siguiente muestra el diseño de clase para el lado servidor.

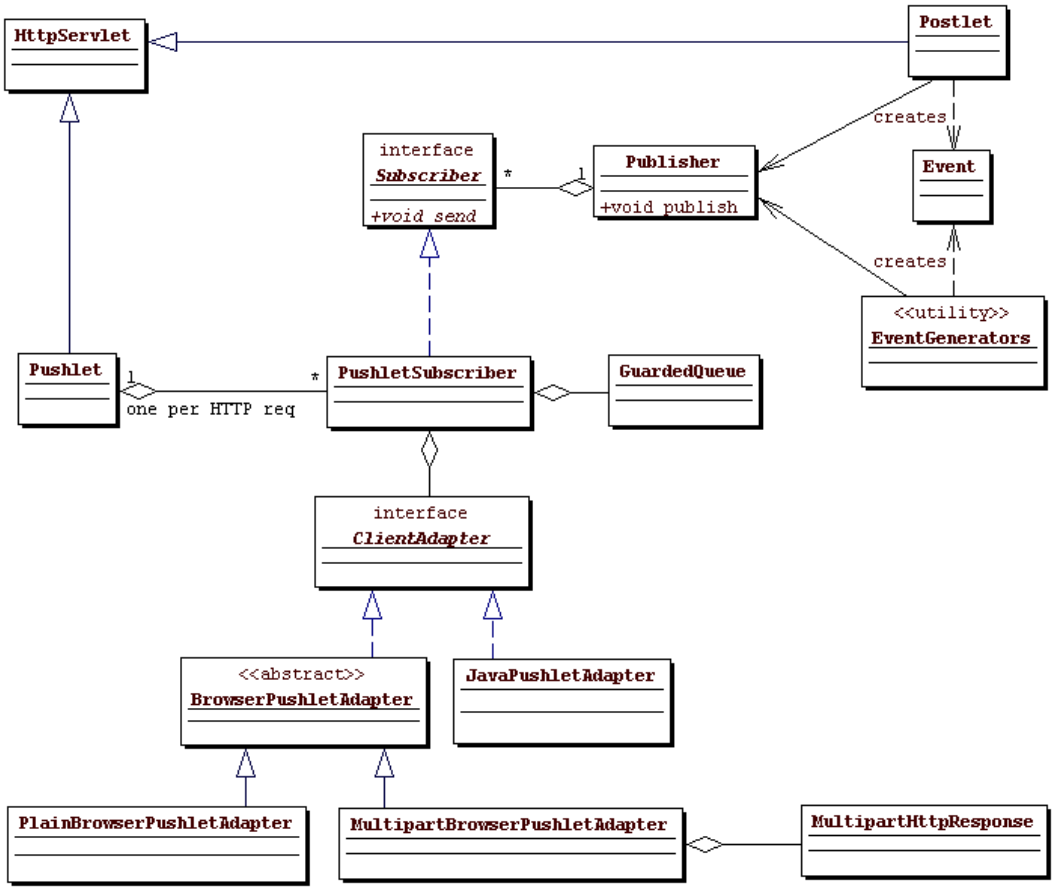

Figura 2 6. (Just van Broecke, 2002)

Diagrama de clase para el framework Pushlet

La siguiente figura muestra el diagrama de secuencia para un cliente subscribiendo para eventos del Publisher.

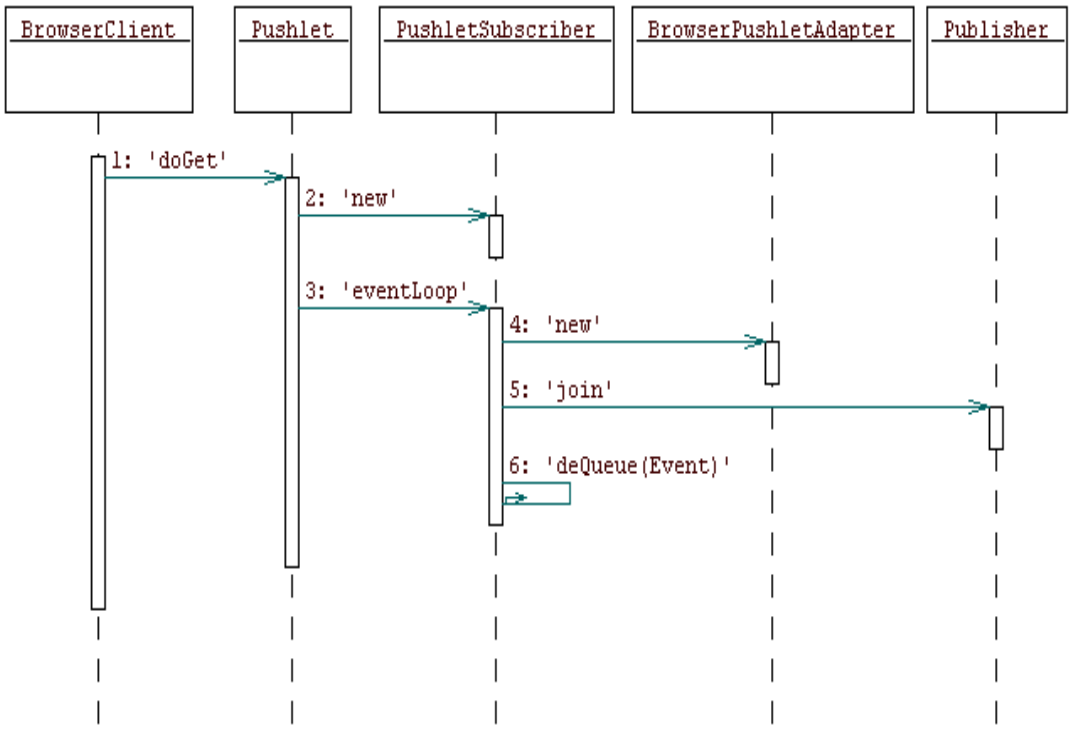

Figura 2 7

Diagrama de secuencia

El Pushlet está involucrado con el método doGet (). Porque múltiples clientes pueden invocar el mismo objeto de Pushlet, pero el mismo objeto no debe ser un suscriptor. En lugar de eso, se delegan todas las suscripciones (y posteriormente el manejo de eventos) para crear un nuevo objeto para cada doGet () y corre hasta el final del eventLoop ().

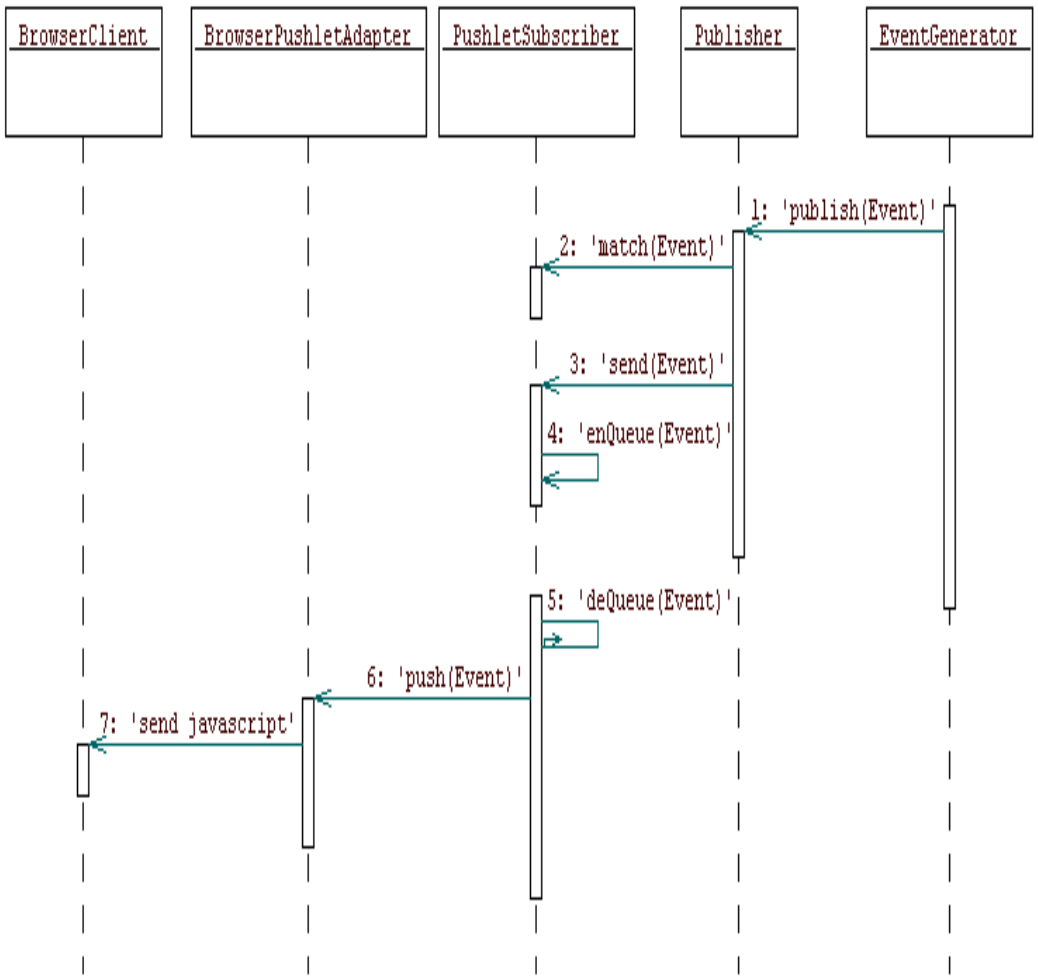

Figura 2 8

Diagrama de secuencia para Publicar

2.4.5 JQuery

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

jQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX. La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX.

jQuery es método sencillo que no necesita de escribir mucho código. El siguiente es un ejemplo de cómo refrescar un "div" cada diez segundos sin recargar la página completa.

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"
></script>

<script type="text/javascript">

var url = 'servlet/util.GetDisplayData'

$(document).ready(function() {

$.ajaxSetup({ cache: false });

setInterval(function() {$("#displayarea").load(url); }, 10000);

});

</script>
```

En este ejemplo la función "*load*" carga los datos del servidor y los coloca en el documento HTML en el elemento seleccionado, en este caso el *tag* "div" que se llama "#displayarea".

2.5 Conclusiones parciales

El análisis de las diferentes herramientas presentadas fue realizado con base en los siguientes criterios, a fin de determinar cuál es la óptima para implementar en el sitio "chess.uclv.edu.cu".

- **Apoyo -** la herramienta debe tener variados módulos que permitan el manejo de diferentes funciones y aplicaciones.
- **Facilidad de usar –** la herramienta debe ser fácil de entender y manipular.
- **Flexibilidad –** la herramienta debe poseer funcionalidades que le permitan cumplir con el objetivo del trabajo.

Luego del análisis planteado, se valora que de las herramientas usadas, Node.js demostró mejor las cualidades necesarias y por eso se propone implementar esta herramienta en la aplicación chess.uclv.edu.cu. A continuación se explora cómo realizar la implementación de esta herramienta y garantizar la funcionalidad de acuerdo con el objetivo del trabajo. : jugar en tiempo real un juego de ajedrez en el sitio chess.uclv.edu.cu.

Capítulo III

- Implementación y prueba

3.1 Introducción

Este capítulo demuestra el uso de Node.js en una aplicación de ajedrez. Se usa el sitio web chess.uclv.edu.cu en combinación con los módulos de Node.js.

3.2 Sistema de registro y autenticación

El sistema cuenta con un mecanismo de registro y autenticación. Los usuarios no registrados tienen que registrarse si quieren navegar dentro el sitio como un usuario registrado con más funcionalidades disponibles. Una vez que sean usuarios registrados pueden cambiar su contraseña, modificar y eliminar su cuenta.

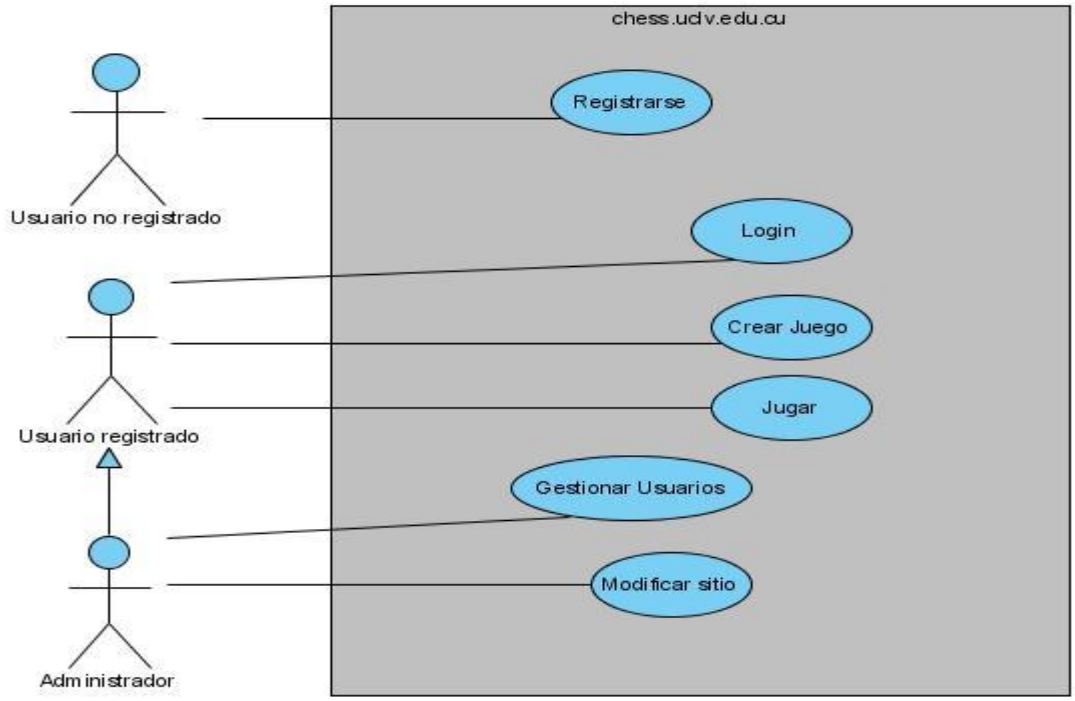

Figura 3 1

Diagrama de caso de uso del sistema de registro

3.3 Descripción de Node.js

El node.js es una aplicación avanzada de JavaScript y tiene muchos ficheros y módulos. Los 4 ficheros que vamos a utilizar son los siguientes:

- App.js
- Chessclient.js
- Chesserver.js
- Server.js

El uso de los módulos está basado de los requerimientos dela aplicación web, cada módulo tiene un propósito específico y solamente puede hacer una tarea determinada de antemano. Cuando alguien va a crear una aplicación web en tiempo real, tiene que saber exactamente qué quiere y cuál módulo puede ofrecerle lo que quiere, para saber cuáles serían esos módulos, es necesario investigar cada uno de ellos.

Luego de concluida la investigación, se obtuvieron como resultado los siguientes cuatro módulos, como los más a propósito para conseguir los resultados esperados:

- Socket.io
- Mysql
- Request
- Forever

3.4 Configuración de Los Ficheros de JavaScript

App.js

Este fichero tiene toda la configuración del sitio, como se observa en la figura 3 2, el parámetro chessserver es el otro fichero JavaScript necesario, el url, el puerto donde se recepcionan los cambios y la conexión con la base de datos en MySQl.

```
'use strict';

var ChessServer = require('./chessserver'),
    config = {server_url: 'http://www.chess.local',
        server_port: 8088,
        online_player_refresh_interval: 30000,
        // db config
        dbhost: 'localhost', dbuser: 'root', dbpassword: 'ok', database: 'chess'},
    games = {};
```

Figura 3 2

Aquí en la figura 3 3 se muestran el/los módulo(s) que se necesitan. Está poniendo en funcionamiento (*calling*) el fichero socket.io.

```
var chess_server = ChessServer.create(config),
    io = require('socket.io').listen(config.server_port);

io.sockets.on('connection', function (socket) {
```

Figura 3 3

En figura 3 4 se actualizan los jugadores que están conectados al sitio.

C:\Users\BOUNCE\Desktop\work\www\realtime\app.js - Notepad++

```
15
16      // push online player to clients, burden presence server implementation
17      setInterval(function () {
18          chess_server.get_online_players(function(data){
19              socket.emit("update online players", data);
20          });
21      }, config.online_player_refresh_interval);
22
```

JavaS length : 3108 lines : 82 Ln : 1 Col : 1 Sel : 0 | 0 UNIX ANSI INS

Figura 3 4

En la figura 3 5, el socket está esperando un cambio de estado, si el jugador mueve una pieza, su movimiento será almacenado en la BD.

C:\Users\BOUNCE\Desktop\work\www\realtime\app.js - Notepad++

```
54
55      // on player move
56      socket.on('player move', function (data) {
57          socket.get('session', function (err, session) {
58              var opponent_socket,
59                  request = require('request');
60
61              if (session.color == 'White') {
62                  var opponent_socket = games[session.game].black.socket;
63                  opponent_socket.emit('update board', data);
64              } else {
65                  var opponent_socket = games[session.game].white.socket;
66                  opponent_socket.emit('update board', data);
67              }
68              // handle the move, store in db, etc...
69              request.post(
70                  config.server_url + '/chessajaxied.php?gameid=' + session.game,
71                  {form: {chessmove: data.move_str, player: session.player}},
72                  function (error, response, body) {
73                      if (!error && response.statusCode == 200) {
74                          console.log(body);
75                          opponent_socket.emit('update move description', body);
76                      }
77                  }
```

JavaSc length : 3108 lines : 82 Ln : 1 Col : 1 Sel : 0 | 0 UNIX ANSI INS

Figura 3 5

Chessserver.js

Tiene los parámetros de la conexión de la base datos. Figura 3 6 se muestra.

```
C:\Users\BOUNCE\Desktop\work\www\realtime\chessserver.js - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
chessserver.js
20   * player functionality, a burden presence functionality to get a clue of which
21   * are playing chess at that moment.
22   */
23  ChessServer.prototype.get_online_players = function (callback) {
24      var mysql = require('mysql'),
25          result = false;
26      var connection = mysql.createConnection({
27          host     : this.config.dbhost,    // localhost
28          user     : this.config.dbuser,     // root
29          password : this.config.dbpassword, // ok
30          database : this.config.database // chess
31      });
32
33      connection.connect();
JavaScript fi length : 1675  lines : 54  Ln : 30  Col : 49  Sel : 0 | 0  UNIX  ANSI  INS
```

Figura 3 6

En la figura 3 7 se muestra la consulta (*query*).

```
C:\Users\BOUNCE\Desktop\work\www\realtime\chessserver.js - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
chessserver.js
34
35      connection.query("select pl_identifier, pl_score_points"
36          + " from mcc_player"
37          + " where pl_is_active = 1"
38          + "       and pl_is_validated = 1"
39          + "       and pl_email_address is not null"
40          + "       and ( unix_timestamp( now() ) - unix_timestamp(pl_last_check) ) <=
            60000 / 1000"
41          + " order by pl_identifier desc", function(err, rows, fields) {
42
43          if (err) throw err;
44          callback(rows);
45      });
46
JavaScript fi length : 1675  lines : 54  Ln : 30  Col : 49  Sel : 0 | 0  UNIX  ANSI  INS
```

Figura 3 7

Chessclient.js

En este fichero pasa toda la "acción", aquí se declaran los parámetros del juego, los jugadores, el id del juego, configuración, movimiento, módulo.

Se muestra en las Figuras 3 8 y 3 9.

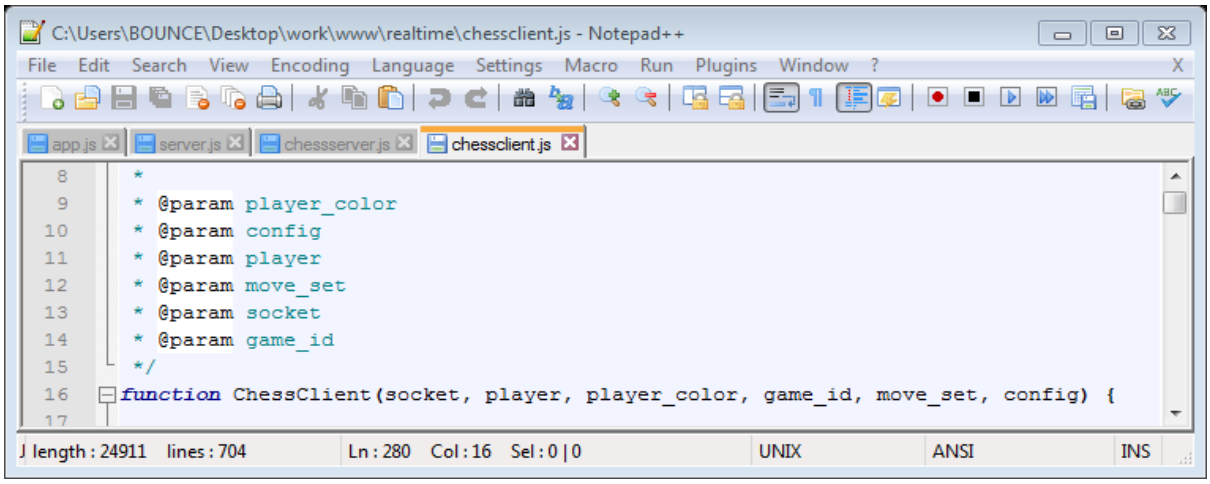

Figura 3 8

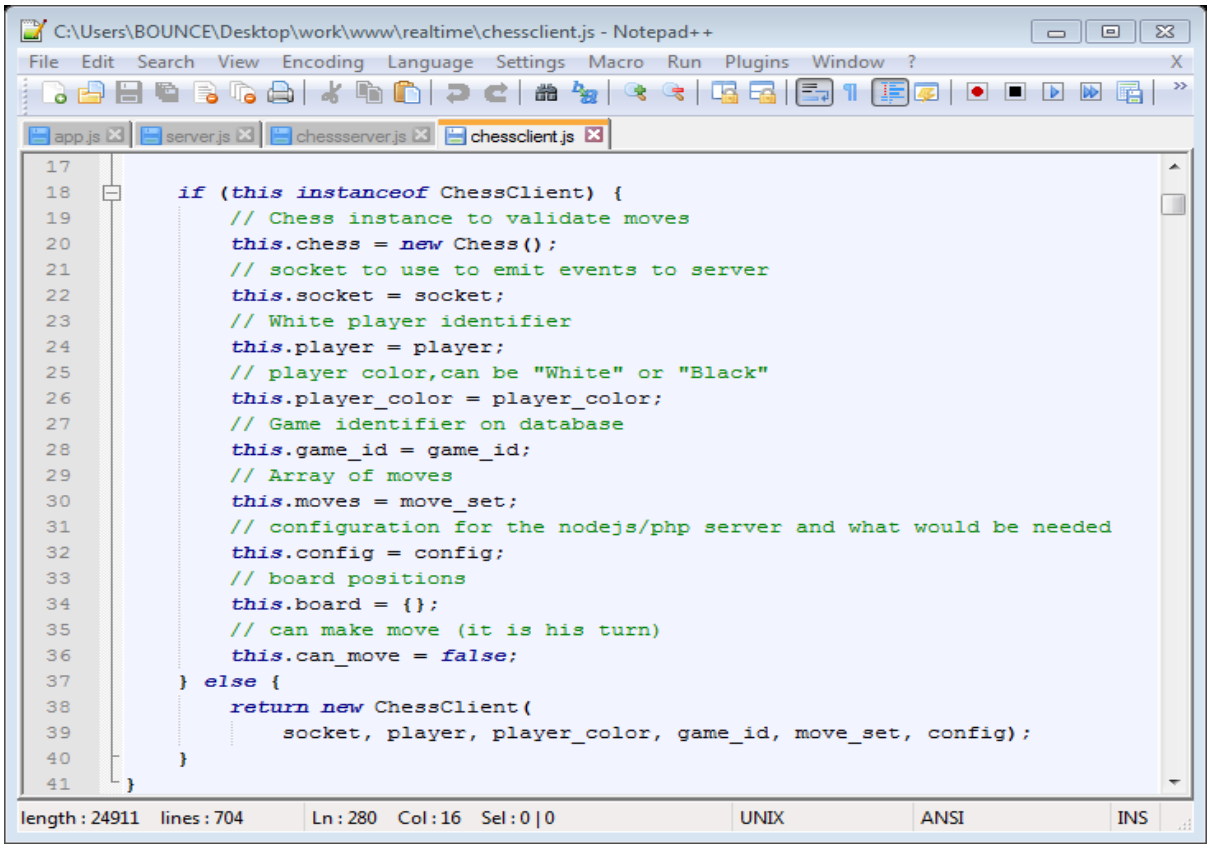

Figura 3 9

El fichero chessclient.js interactúa son el servidor node.js, almacena la lista de movimientos en la base de datos y actualiza el tablero de ajedrez, actualiza los jugadores y los juegos. Básicamente es el enlace entre el servidor e interfaz usuario del sitio de ajedrez.

Server.js

En la figura 3 10 está activando el módulo que se llama *forever*.

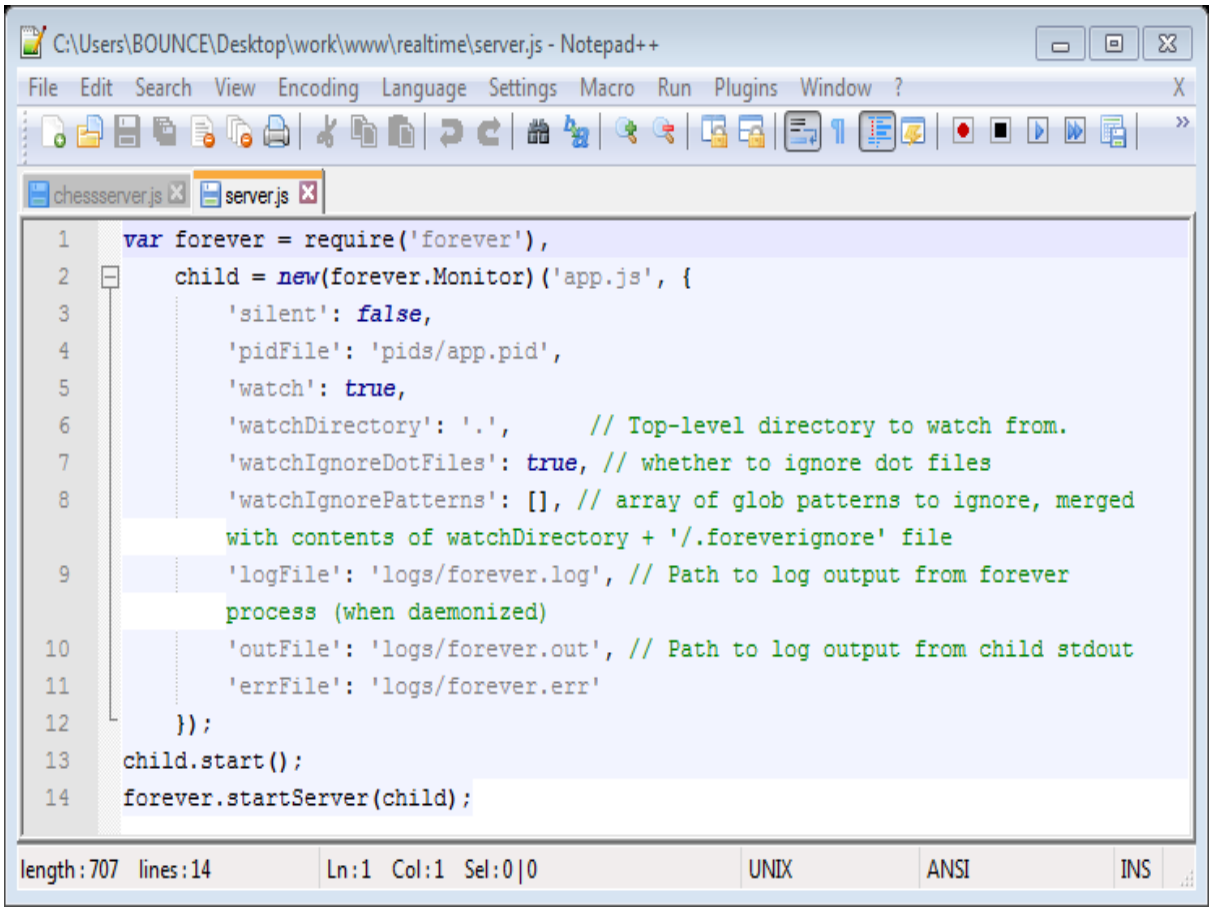

Figura 3 10

3.5 Configuración de las Carpetas

Configuración del servidor Xampp

Xampp tiene un servidor de Apache y MySQL. Es posible con node.js crear un servidor propio, no hace falta el xampp, esa es otra ventaja de node.js, pero ya el sitio montado en xampp está funcionando bien, además se está utilizando el MySQL, servidor que provee el xampp para la conexión con la base de datos.

En la Figura 3.2.1 se muestra el panel del control de xampp, como se observa, los servidores de Apache y MySQL están corriendo en los puertos 80, 443 y 3306. Aquí se pueden cambiar los puertos o cualquier otra configuración deseada.

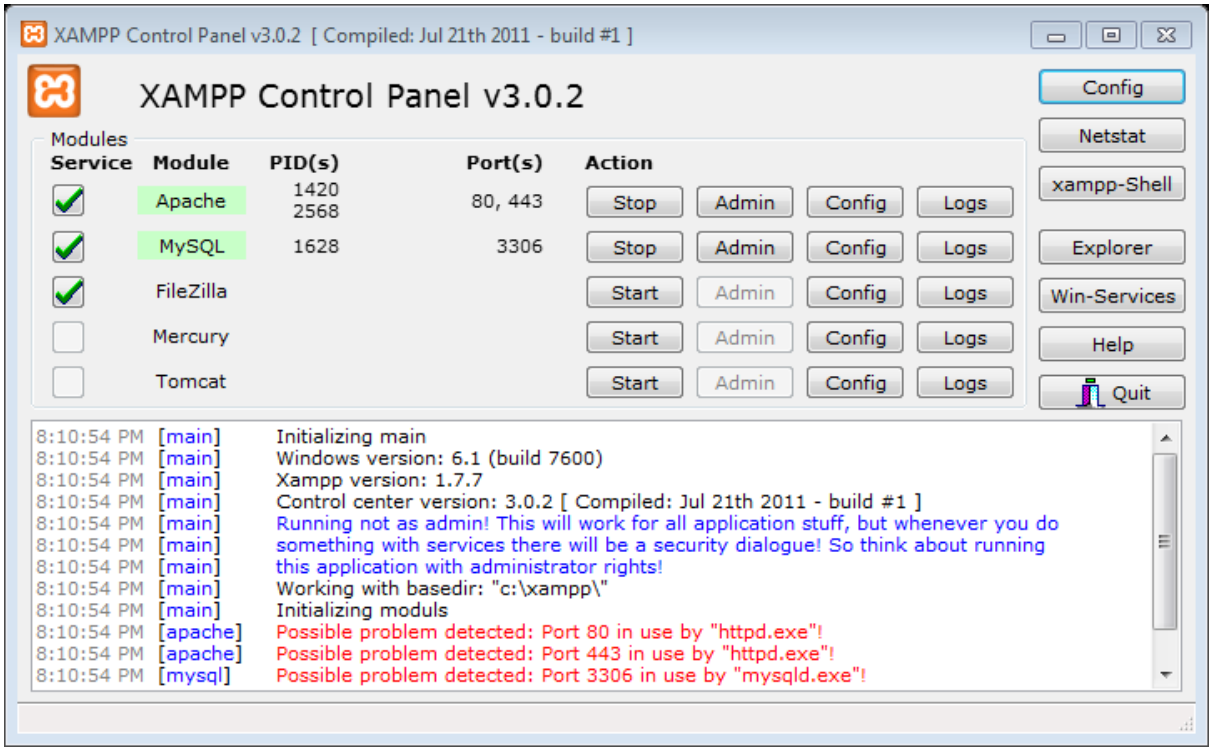

Figura 3 11

La Figura 3 11 está mostrando la dirección del sitio.

Los 4 ficheros de JavaScript están en una carpeta con los módulos, la carpeta se llama “realtime”, la misma se encuentra situada dentro de la carpeta del juego que se llama

"www". La carpeta "www" está adentro de otra carpeta que se llama "htdocs", "htdocs" es una carpeta fundamental en el servidor xampp.

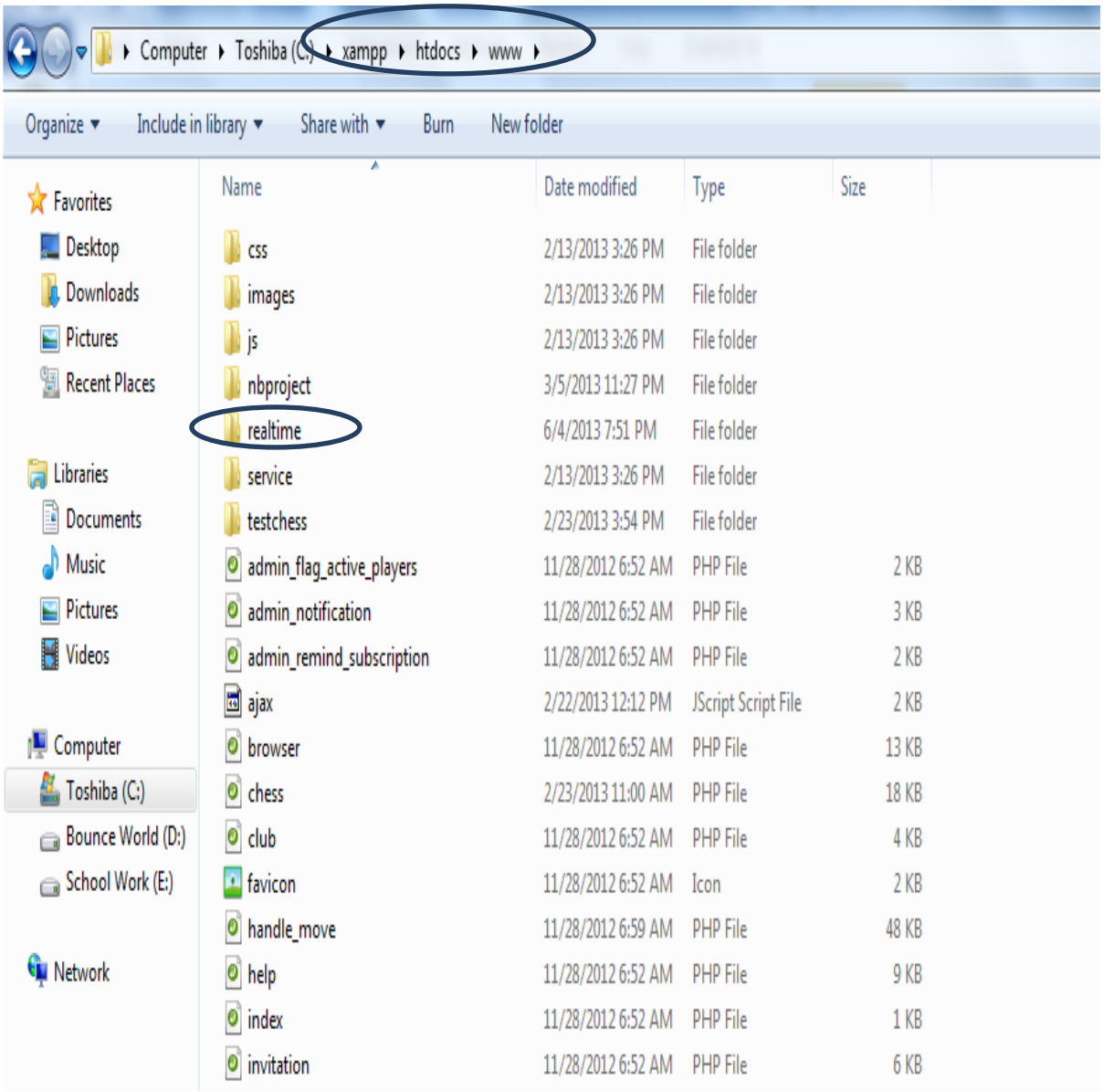

Figura 3 12

3.6 Configuración de los módulos

Dentro de la carpeta "realtime" los ficheros creados y una carpeta que se llama "node_modules", dentro de esa carpeta están los módulos que se utilizarán.

La Figura 3 13 muestra la carpeta donde están los módulos.

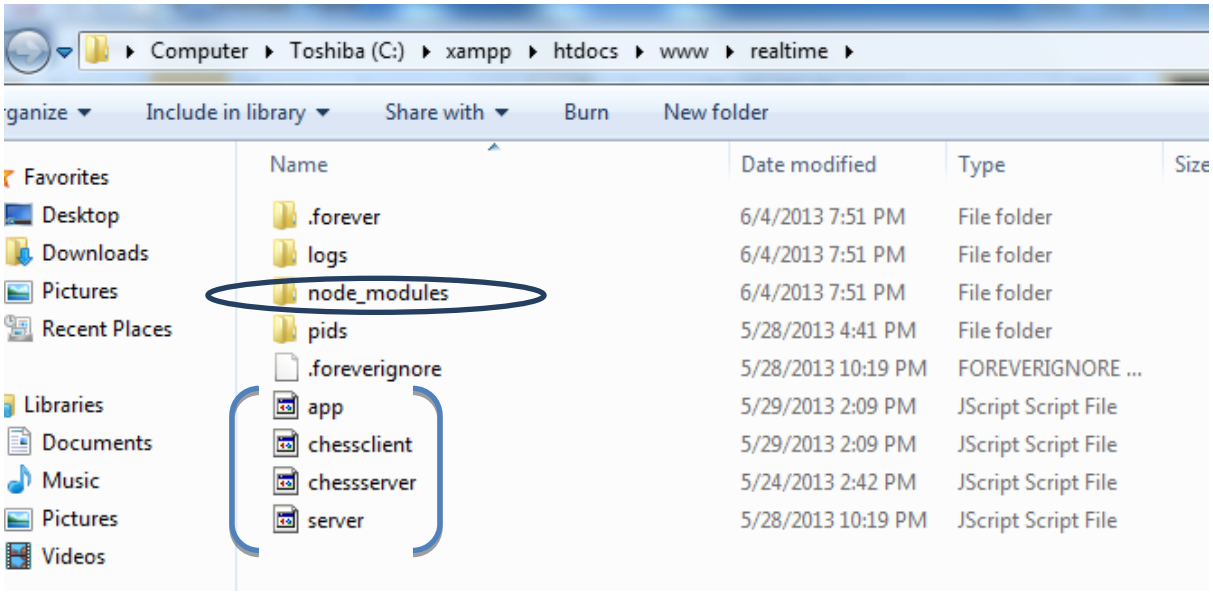

Figura 3 13

Aquí están los módulos. Figura 3 14 los muestra a continuación.

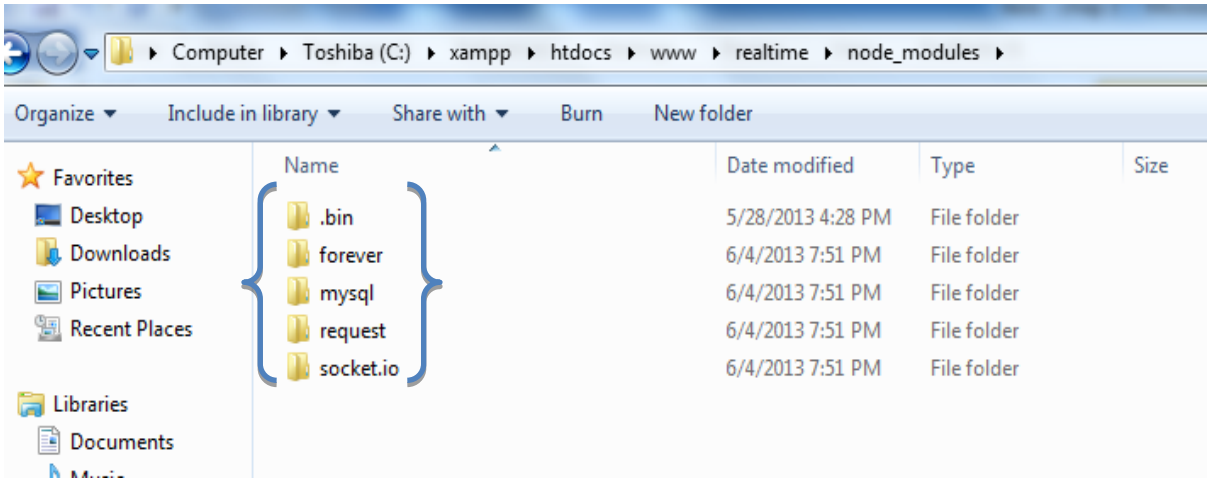

Figura 3 14

3.6.1 Forever

Este módulo es responsable de que los scripts sigan corriendo infinitamente. Tiene dos carpetas más importantes, una se llama “lib” que es la biblioteca y una carpeta llamada “node_modules” también, dentro esa carpeta están módulos pequeños que forman el

módulo *forever* y son responsables de todo el funcionamiento. Dentro de la biblioteca hay un fichero de JavaScript que se llama *forever*, aquí todas las variables están declaradas, los ficheros que se necesitan están activados, básicamente es la página que incorpora todo. En la Figura 3 15 se muestran las carpetas.

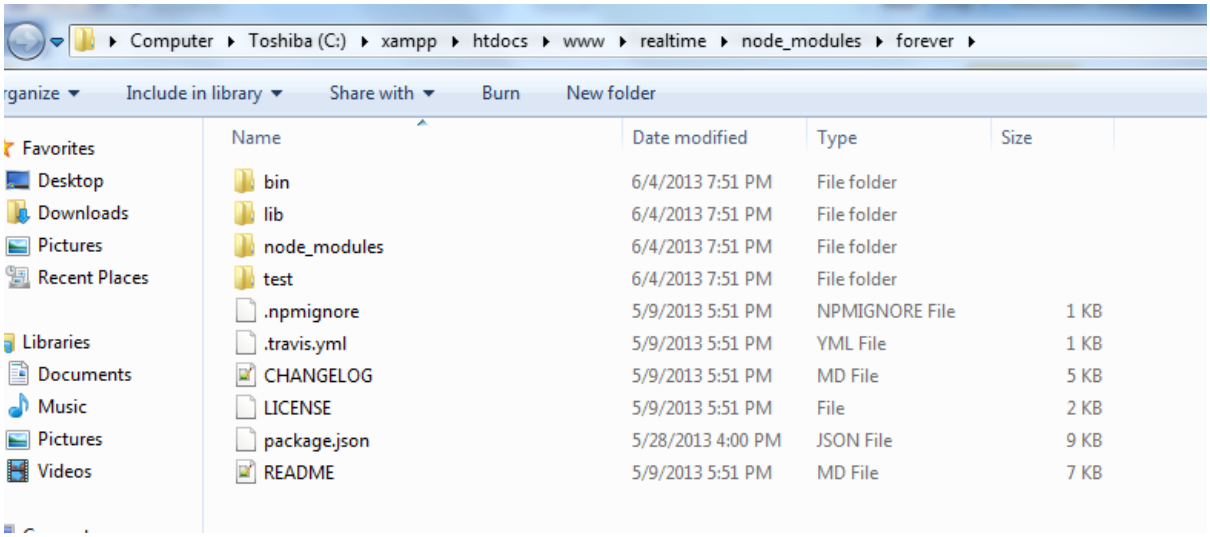

Figura 3 15

En la Figura 3 16 se muestra el fichero forever.js y adentro la biblioteca.

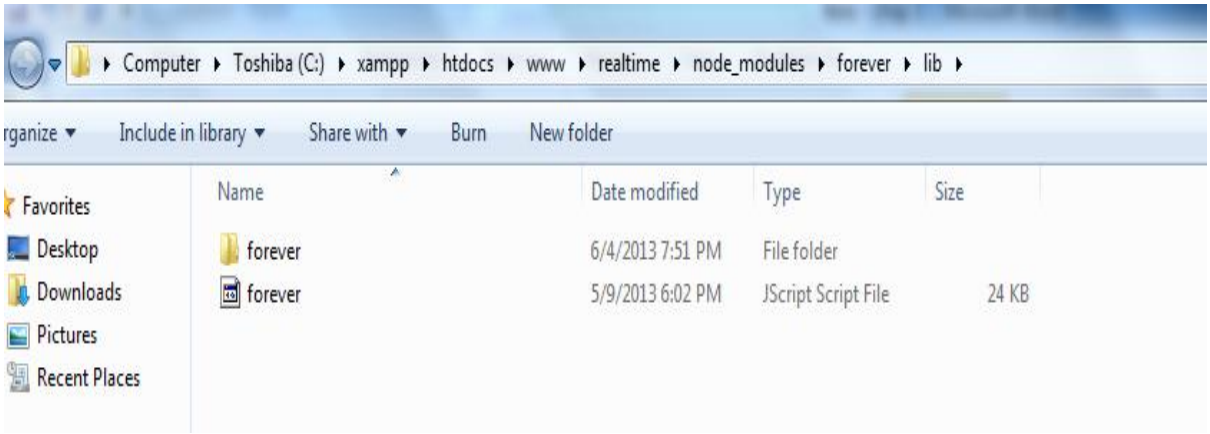

Figura 3 16

3.6.2 Request

Este módulo facilita llamadas de HTTP (*Hyper Text Transfer Protocol*), apoya HTTPS y los devuelve por defecto. Las llamadas HTTP son solicitudes desde el cliente al servidor. Cuando el cliente, en ese caso el jugador, hace algo, puede ser un movimiento de las piezas de ajedrez, realiza una búsqueda o *logout*, envía una solicitud al servidor y el servidor procesa la solicitud y devuelve una respuesta por defecto.

Figura 3 17 muestra los ficheros de ese módulo, también tiene una carpeta llamada "node_modules" que es la responsable de la ejecución de las peticiones, no tiene ningún fichero principal. Todos los ficheros tienen un propósito específico y funcionan como una unidad.

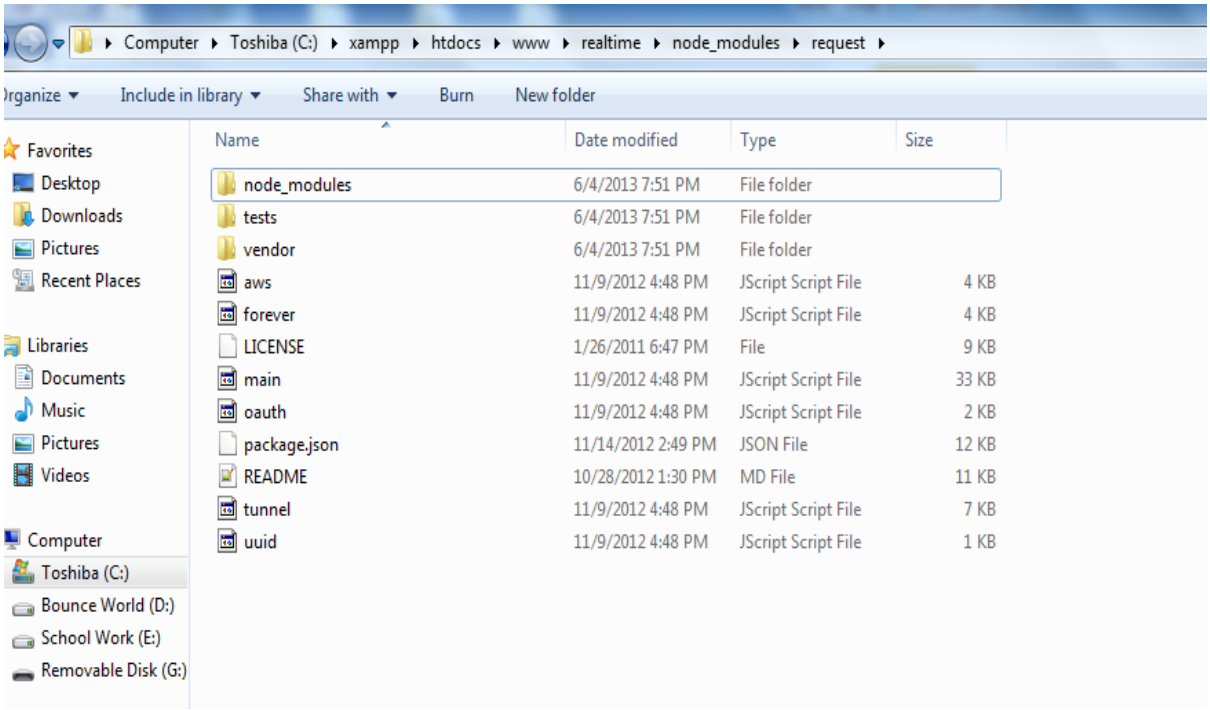

Figura 3 17

3.6.3 MySQL

Este módulo establece la conexión con el sitio web y la base de datos. Ejecuta las consultas (queries) solicitadas por los clientes (jugadores) y actualiza las tablas en la base de datos. Tiene una biblioteca que es responsable por la conexión. Hay un fichero que se llama index.js, este es el fichero principal y activa todos los demás ficheros que se necesitan.

La Figura 3 18 está mostrando los contenidos del módulo.

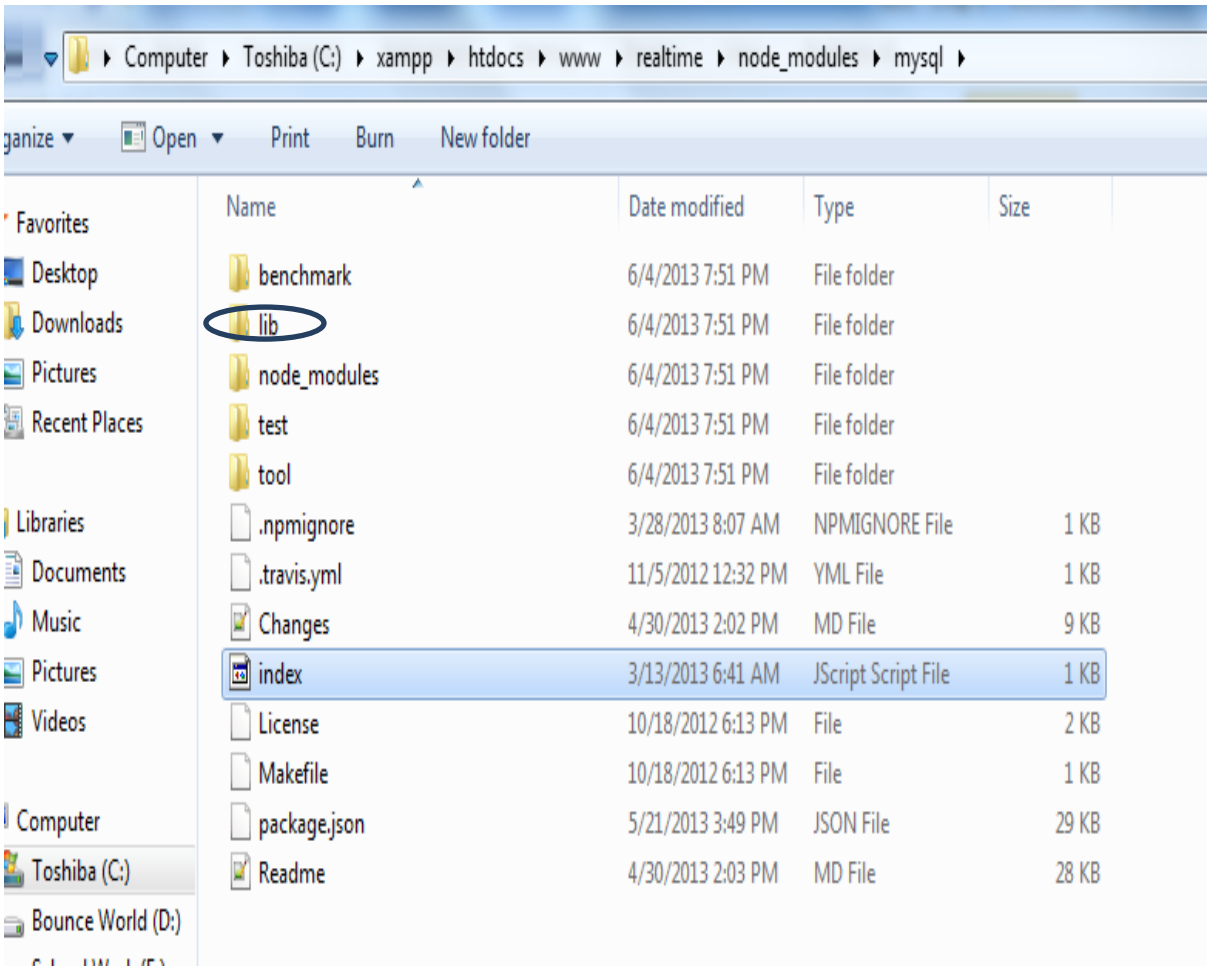

Figura 3 18

Aquí se presentan el contenido index.js y los ficheros que necesitan. Figura 3 19.

```
index.js
1   var Connection       = require('./lib/Connection');
2   var ConnectionConfig = require('./lib/ConnectionConfig');
3   var Types            = require('./lib/protocol/constants/types');
4   var SqlString        = require('./lib/protocol/SqlString');
5   var Pool             = require('./lib/Pool');
6   var PoolConfig       = require('./lib/PoolConfig');
7
8   exports.createConnection = function(config) {
9     return new Connection({config: new ConnectionConfig(config)});
10  };
```

Figura 3 19

En la Figura 3 20 se señalan los contenidos de la carpeta "lib" o la biblioteca del módulo MySQl. El fichero más importante es el "ConnectionConfig", aquí se configura la conexión de la base de datos.

Computer ▸ Toshiba (C:) ▸ xampp ▸ htdocs ▸ www ▸ realtime ▸ node_modules ▸ mysql ▸ lib ▸

anize ▾ Include in library ▾ Share with ▾ Burn New folder

Favorites
Desktop
Downloads
Pictures
Recent Places

Libraries

Name	Date modified	Type	Size
protocol	6/4/2013 7:51 PM	File folder	
Connection	4/5/2013 6:38 PM	JScript Script File	5 KB
ConnectionConfig	4/30/2013 12:46 PM	JScript Script File	4 KB
Pool	3/13/2013 6:41 AM	JScript Script File	5 KB
PoolConfig	3/13/2013 6:41 AM	JScript Script File	1 KB

Figura 3 20

La Figura 3 21 muestra el contenido de la página "ConnectionConfig", aquí se puede modificar el host, el puerto, nombre de la base de datos, la contraseña y muchas más opciones.

```
module.exports = ConnectionConfig;
function ConnectionConfig(options) {
  if (typeof options === 'string') {
    options = ConnectionConfig.parseUrl(options);
  }

  this.host               = options.host || 'localhost';
  this.port               = options.port || 3306;
  this.socketPath         = options.socketPath;
  this.user               = options.user || undefined;
  this.password           = options.password || undefined;
  this.database           = options.database;
  this.insecureAuth       = options.insecureAuth || false;
  this.supportBigNumbers  = options.supportBigNumbers || false;
  this.bigNumberStrings   = options.bigNumberStrings || false;
  this.debug              = options.debug;
  this.timezone           = options.timezone || 'local';
  this.flags              = options.flags || '';
  this.queryFormat        = options.queryFormat;
  this.pool               = options.pool || undefined;
  this.multipleStatements = options.multipleStatements || false;
  this.typeCast           = (options.typeCast === undefined)
    ? true
    : options.typeCast;
```

Figura 3 21

3.6.4 Socket.io

Este es el módulo responsable de la ejecución del tiempo real, tiene una biblioteca de lado servidor que funciona con node.js y una de lado cliente que funciona con los navegadores. Socket.io le permite a un programador programar el lado cliente y el lado servidor en JavaScript. Tiene un fichero principal socket.io.js que es responsable de conectar el cliente con el servidor. Este módulo es manejado por eventos.

Esto significa que cuando un evento ocurre en el lado cliente, automáticamente se envía un mensaje al lado servidor, el lado servidor analiza ese evento para saber qué cambios ha invocado sobre el cliente y después manda una respuesta a los clientes afectados por ese evento.

Ejemplo:

Si un jugador de color blanco hace un movimiento e1-e4, este es un evento o un cambio de datos, el tablero de ese jugador está actualizado, él sabe que su turno terminó

pero su oponente no. Aquí viene el uso del socket.io, el oponente con el color negro está esperando porque no sabe que es su turno. Este evento llega al servidor, el servidor analiza el evento y desde ese evento sabe el id del juego, estado del juego y estado del jugador. El servidor actualiza las páginas de todos los clientes (jugadores) afectados por ese cambio. Ahora el oponente sabe que es su turno, este proceso funciona en milisegundos, los movimientos se ven inmediatamente.

Lado servidor del socket.io

Figura 3 22 los contenidos del socket, la primera biblioteca que se ve es el lado servidor.

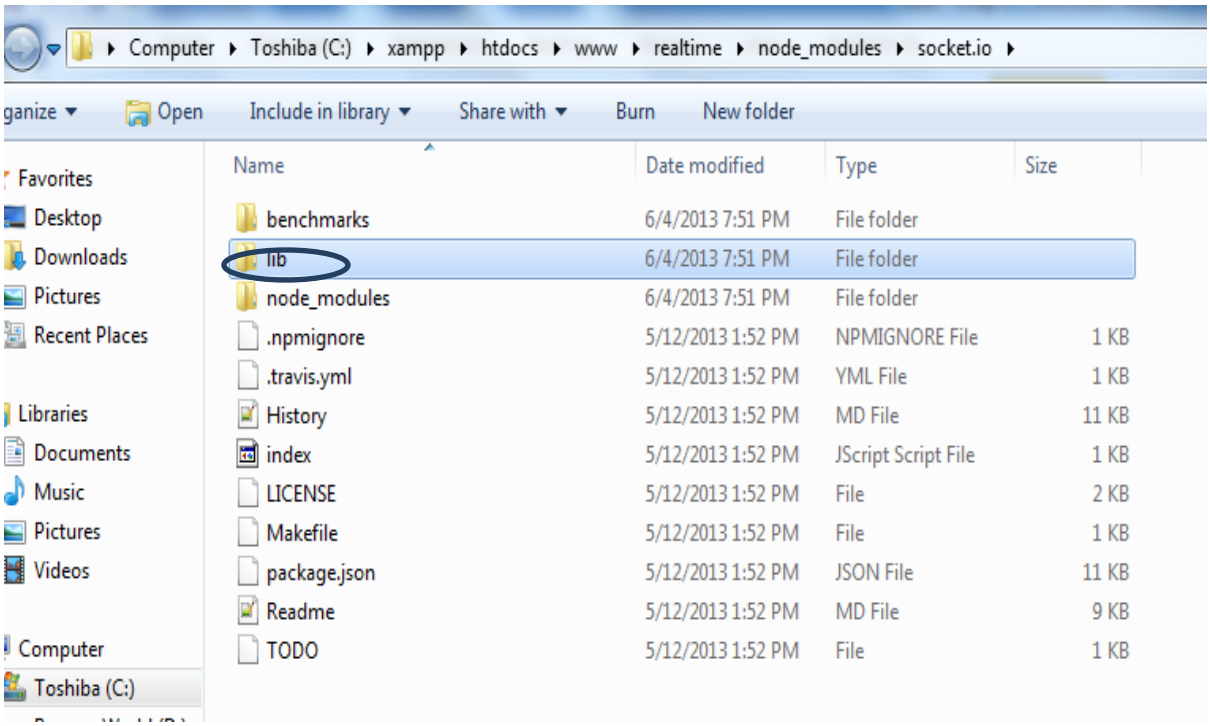

Figura 3 22

La Figura 3 23 muestra los contenidos dentro de la biblioteca del lado servidor. Aquí se ve el fichero socket.io.js. Este fichero activa todos los ficheros mostrados abajo, cada uno de esos ficheros tiene un propósito específico.

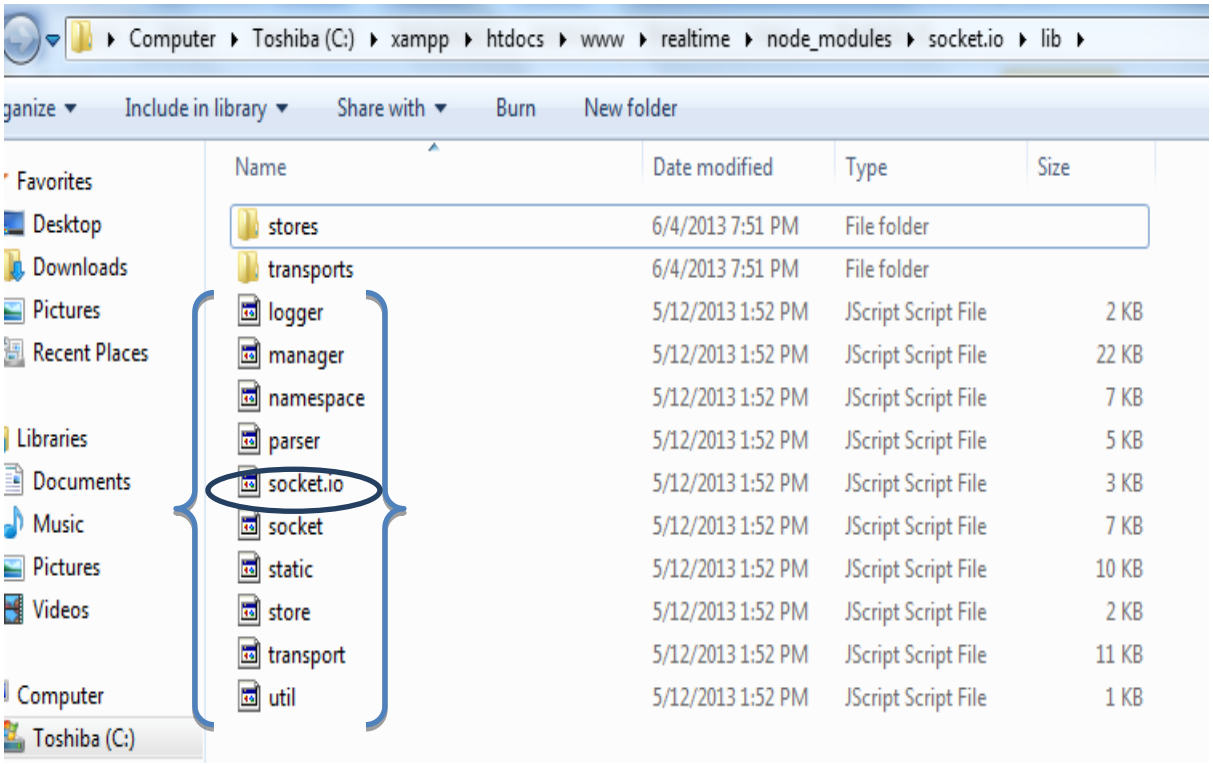

Figura 3 23

Aquí en la Figura 3 24 socket.io está activando el lado cliente, la carpeta socket.io-client.

```
var client = require('socket.io-client');

/**
```

Figura 3 24

Aquí en la Figura 3 25 está activando todos los ficheros que necesita.

```
exports.Manager = require('./manager');

/**

exports.Transport = require('./transport');

/**

exports.Socket = require('./socket');

/**

exports.Static = require('./static');

/**
```

Figura 3 25

Lado cliente del socket.io

Tiene también una carpeta dentro que se llama “node_modules”, dentro de esa carpeta está la carpeta socket.io-client que es responsable por el lado cliente.

La Figura 3 26 lo muestra.

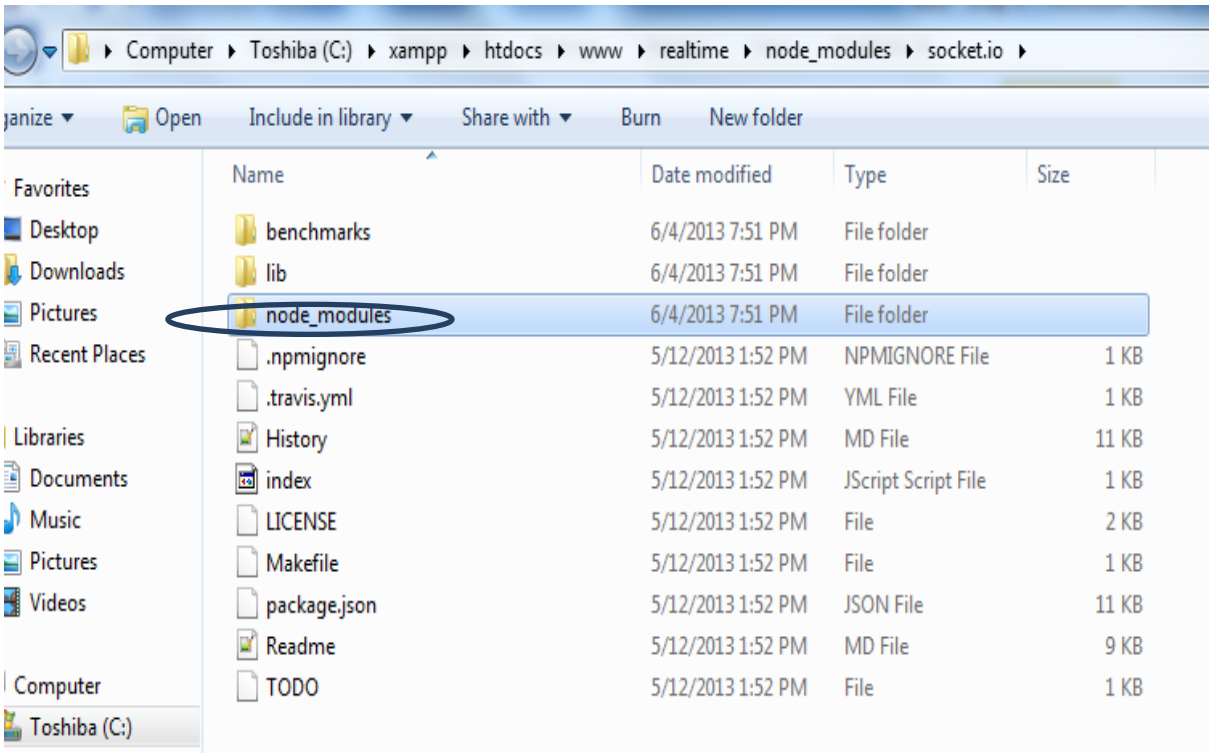

Figura 3 26

Aquí se ve la carpeta “socket.io-cliente”. Figura 3 27

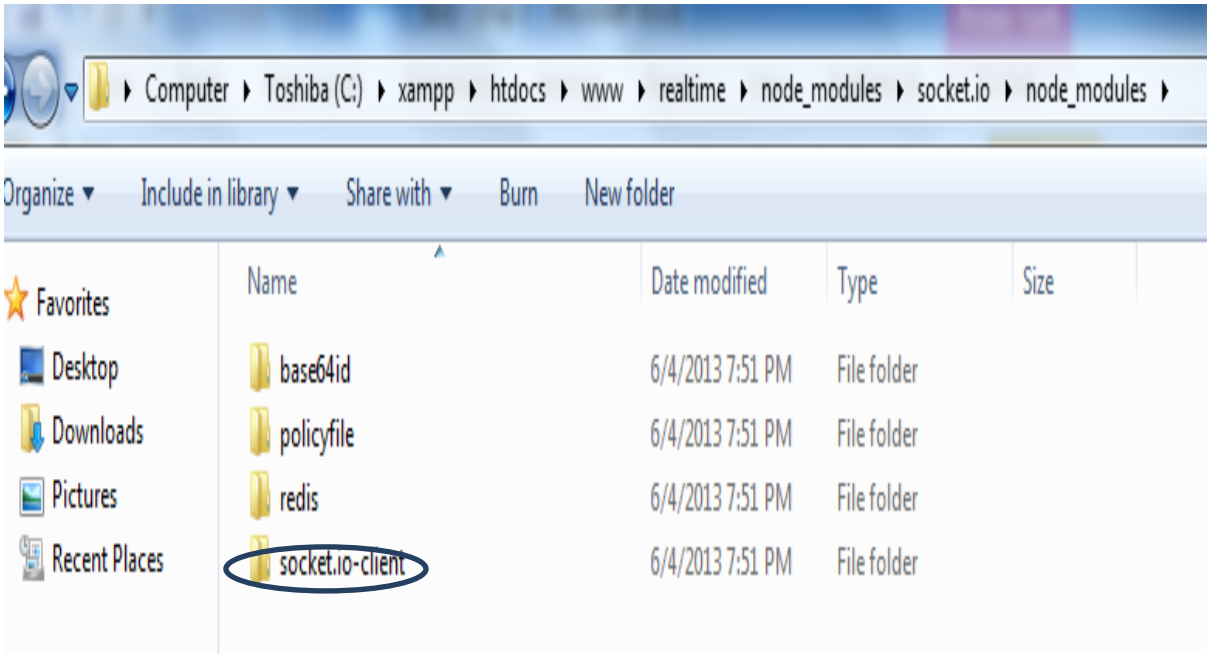

Figura 3 27

La carpeta socket.io cliente tiene su biblioteca también, está señalada en la Figura 3 28

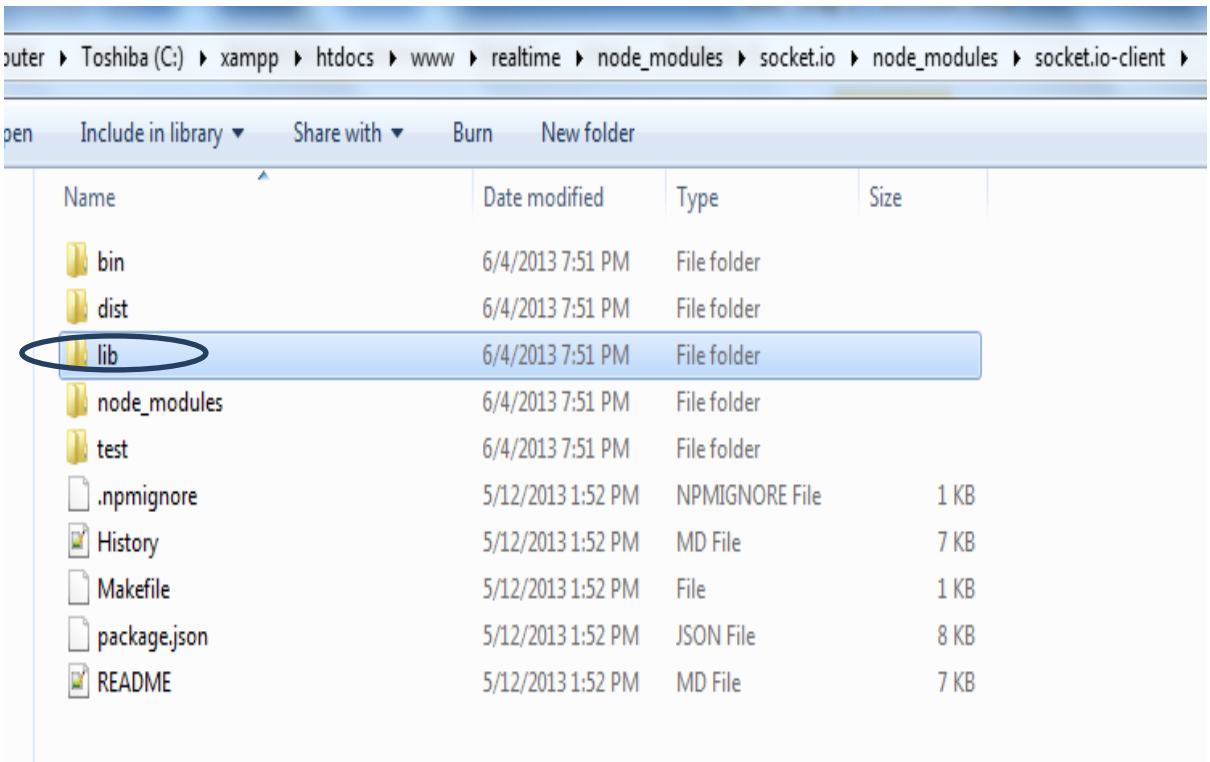

Figura 3 28

Dentro de la biblioteca hay un fichero se llama socket, el cual maneja toda la información, activando los ficheros que se necesitan, escucha al puerto y está conectado

con el lado servidor socket.io.js. Hay un fichero events.js que es responsable por todos los eventos (cambios) que ocurren en el lado cliente.

La Figura 3 29 muestra este fichero y los demás ficheros importantes.

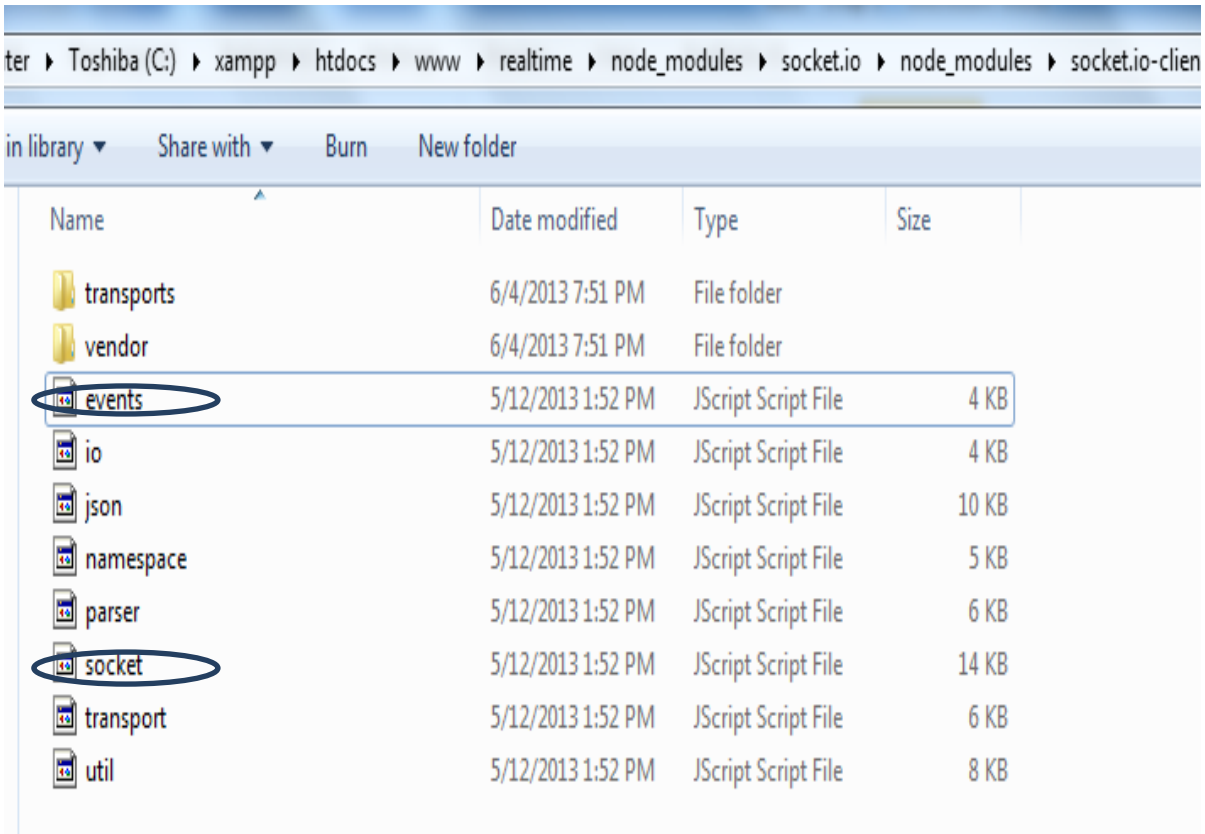

Figura 3 29

3.7 Arrancando el sitio

- Abra el *command prompt* interfaz, puede teclear *cmd* en el *Run*.
- Teclee cd, copia y pegue la dirección de la carpeta “realtime”.
- Presione el botón “Enter”
- Escriba “node app.js” y presione “Enter”
- Iniciar el navegador, escriba localhost/www o puede escribir localhost:8088

Figura 3 30 Interfaz de *command prompt*

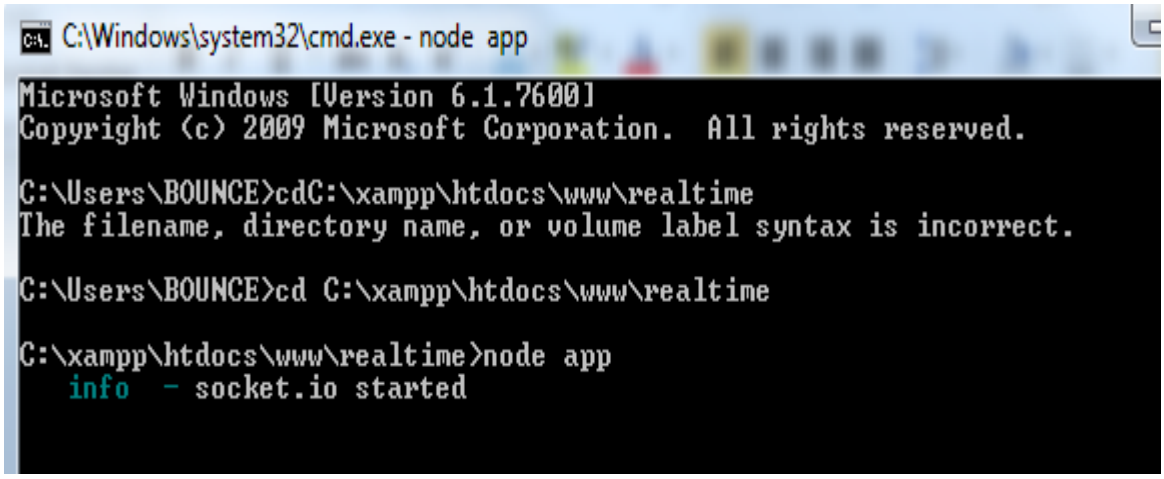

figura 3 30

La Figura 3 31 muestra el sitio montado en el navegador. La página de autentificación.

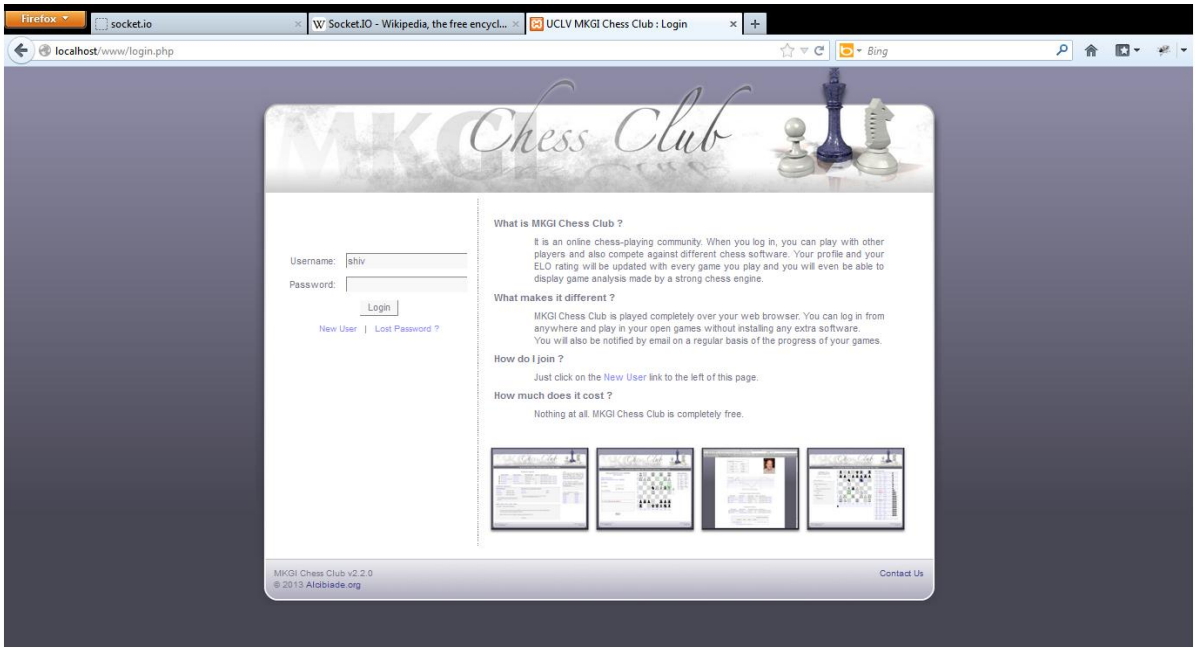

Figura 3 31

Hay un ejemplo con dos jugadores jugando. Cada uno ve cuando el otro hace su movimiento.

Figura 3 32 muestra al jugador “shiv” (color blanco) contra el jugador “ice” (color negro)

Figura 3 32

Figura 3 33 muestra el interfaz de Command Prompt, se puede ver socket.io escuchando el puerto, se ven los usuarios que están conectados al sitio.

```
{ game_id: 3, player: 'juancg', opp_timer: 0, own_timer: 7000 }
  debug - websocket writing 5:5+::{"name":"game over timeout","args":[{"game_id":3,"player":"juancg","opp_timer":0,"own_timer":7
  debug - websocket writing 5:::{"name":"finish"}
  debug - websocket writing 5:::{"name":"finish"}
  debug - sending data ack packet
  debug - websocket writing 6:::2+["ok"]
  debug - websocket writing 5:::{"name":"update online players","args":[[{"pl_identifier":"juancg","pl_score_points":1640},{"pl_
e_points":1345}]]}
  debug - websocket writing 5:::{"name":"update online players","args":[[{"pl_identifier":"juancg","pl_score_points":1640},{"pl_
e_points":1345}]]}
  debug - websocket writing 5:::{"name":"update online players","args":[[{"pl_identifier":"juancg","pl_score_points":1640},{"pl_
e_points":1345}]]}
  debug - websocket writing 5:::{"name":"update online players","args":[[{"pl_identifier":"juancg","pl_score_points":1640},{"pl_
e_points":1345}]]}
  debug - emitting heartbeat for client bwYSKX4w3MjDxEPghLKQ
  debug - websocket writing 2::
  debug - set heartbeat timeout for client bwYSKX4w3MjDxEPghLKQ
  debug - got heartbeat packet
  debug - cleared heartbeat timeout for client bwYSKX4w3MjDxEPghLKQ
  debug - set heartbeat interval for client bwYSKX4w3MjDxEPghLKQ
  debug - emitting heartbeat for client lOAuPOz1t-awo2PQhLKS
  debug - websocket writing 2::
  debug - set heartbeat timeout for client lOAuPOz1t-awo2PQhLKS
  debug - got heartbeat packet
  debug - cleared heartbeat timeout for client lOAuPOz1t-awo2PQhLKS
  debug - set heartbeat interval for client lOAuPOz1t-awo2PQhLKS
  debug - websocket writing 5:::{"name":"update online players","args":[[{"pl_identifier":"juancg","pl_score_points":1640},{"pl_
e_points":1345}]]}
  debug - websocket writing 5:::{"name":"update online players","args":[[{"pl_identifier":"juancg","pl_score_points":1640},{"pl_
e_points":1345}]]}
  debug - websocket writing 5:::{"name":"update online players","args":[[{"pl_identifier":"juancg","pl_score_points":1640}]]}
  debug - websocket writing 5:::{"name":"update online players","args":[[{"pl_identifier":"juancg","pl_score_points":1640}]]}
  debug - websocket writing 5:::{"name":"update online players","args":[[{"pl_identifier":"juancg","pl_score_points":1640}]]}
```

Figura 3 33

3.8 Pruebas del sitio

Figura 3 34 muestra el registro del “Elo” de cada jugador.

28 wins, 10 d

[Home | Help | Logout]

Players Currently Registered

Rank	Name	Games	Wins	Draws	Losses	Rating
1.	robot_gnuchess_07	67	41	15	11	1397
2.	robot_gnuchess_02	91	47	5	39	1359
3.	shiv	52	29	10	13	1355
4.	robot_phalanx_06	33	17	5	11	1343
5.	robot_gnuchess_04	34	21	5	8	1333
6.	alcibiade	49	28	8	13	1331
7.	robot_phalanx_07	49	28	8	13	1331
8.	robot_phalanx_04	32	18	3	11	1330
9.	robot_gnuchess_06	31	19	4	8	1328
10.	robot_gnuchess_01	240	124	9	107	1320
11.	robot_gnuchess_05	25	15	4	6	1315
12.	robot_phalanx_01	124	70	8	46	1308
13.	robot_gnuchess_03	52	28	3	21	1307
14.	ice	52	28	10	14	1307
15.	robot_phalanx_02	82	39	7	36	1297

Page 1 2 >

[Show Only Humans | Show Only Robots | Show Both]

Figura 3 34

Figura 3 35 muestra la función “Search”, aquí se busca un oponente.

Search Games

Location: Open Games

Player:

Player Color: Any Color

Opponent:

Search games

figura 3 35

La figura 3 36 está mostrando jugadores que están jugando en tiempo real.

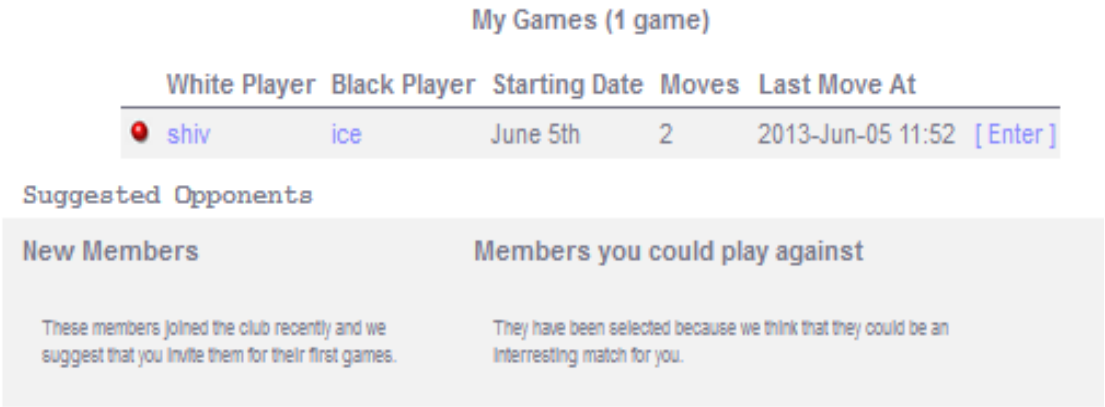

figura 3 36

La figura 3 37 está mostrando el perfil de un jugador donde puede subir foto, cambiar su contraseña y cualquier otra información que desee o requiera modificar.

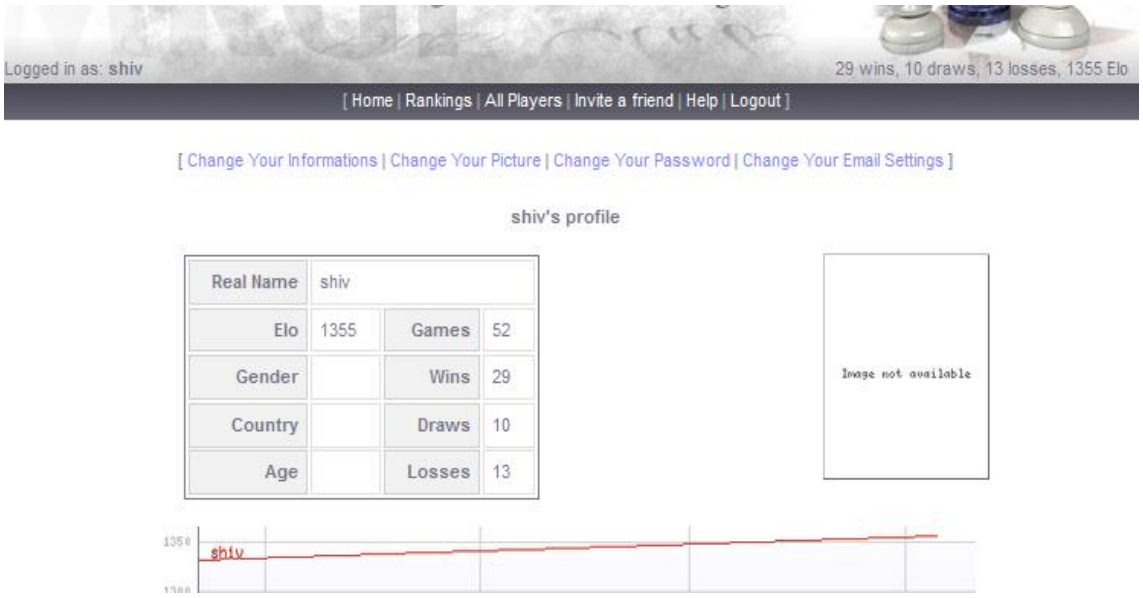

figura 3 37

Conclusiones

Luego del estudio de todo el material disponible y del proceso de análisis y aplicación necesario para llevar a cabo la presente investigación, se pudo arribar a las siguientes conclusiones:

- Se valoraron las plataformas principales investigadas, basados en sus ventajas y desventajas, determinando entonces que la mejor de ellas es Compwebchess_v2.2.
- Se analizaron las herramientas necesarias para desarrollar la solución a la problemática.
- Se estudiaron las tecnologías más populares actualmente en el mundo para interactuar en tiempo real describiendo sus características y cómo se usan.
- Se analizaron patrones, arquitecturas e implementación de distintas tecnologías para encontrar cuál sería la mejor solución a implementar en el sitio de ajedrez de la UCLV (chess.uclv.edu.cu). Se definió que node.js es el mejor para implementar en este sitio.
- Se analizaron los servicios ofrecidos en el sitio web *chess.uclv.edu.cu* buscando sus deficiencias con el objetivo de mejorar esos servicios y añadir más funcionalidad al sitio.
- Se realizó una entrevista al el gran maestro Jesús Nogueira para valorar la investigación y su utilidad en la vida real para mejorar el servicio de ajedrez en la UCLV. El gran maestro sugirió los factores relevantes para crear un sitio web de ajedrez ideal con: una buena interacción entre los jugadores y un sistema de calificación para medir el rendimiento de los mismos.

Las tecnologías de tiempo real son unas de las tecnologías emergentes en el mundo de la programación web. Estas tecnologías pueden ser utilizadas en una gran variedad de áreas, por ejemplo en sitios web que publican los cambios en los precios de la bolsa de valores. Se utilizan también en sitios de chat, para mandar y recibir mensajes rápidamente y en juegos de internet como ajedrez, cartas y billar.

Los usos de esta tecnología son tan versátiles como los medios que se utilizan para implementarla, por eso se considera que la misma es el futuro de la programación web; ya compañías mundialmente reconocidas como Microsoft, Facebook, Yahoo y eBay utilizan tecnologías de tiempo real para exhibir sus productos y servicios al público.

Esta investigación se basa en el funcionamiento e implementación de las tecnologías de tiempo real, así como su utilidad para mejorar el servicio del sitio web de ajedrez en la UCLV.

Recomendaciones

- Adicionar un sistema de chat al sitio que les permita a los jugadores comunicarse entre ellos durante el juego.
- Adicionar diversos sistemas de juego: torneos, por equipos y foros.
- Adicionar juegos de *blitz time* (corto tiempo), son juegos que duran un tiempo definido de antemano, por ejemplo diez minutos.
- Investigar más el node.js, todas las funcionalidades que brindan, sus módulos, *frameworks*, el impacto y alcance sobre el mundo de la programación.

Bibliografía

ADAMS, J. 2006. . *Power Play: The Literature and Politics of Chess in the Late Middle Ages*, Philadelphia: University of Pennsylvania Press.

BECK, K. 1999. *Extreme Programming Explained: Embrace Change*.

BERGSTRÖM, S. 2012. Real Time Multiplayer in HTML5.

BROECKE, J. V. 2002. *Pushlets - Whitepaper* [Online]. Just Objects B.V. [Accessed 20/05/2013 2013].

BROECKE, J. V. D. 2006. *Pushlets Presentation* [Online]. justobjects.nl. [Accessed 20/05/2013 2013].

CALVO, R. Valencia Spain: The Cradle of European Chess.

CHESSQUOTES.COM. 2012. *ChessQuotes.com (Wisdom and Interest)* [Online].

DAVE CRANE, E. P. 2006. *AJAX in Action*, 209 Bruce Park Avenue, Greenwich, CT 06830, Manning Publications Co.

DAVE CRANE, P. M. 2008. Comet and Reverse Ajax: The Next Generation

FIDE. *Laws of Chess* [Online]. [Accessed 2010-08-03.

FOUNDATION, W. 2013. *Comet - Wikipedia, la enciclopedia libre* [Online]. Wikimedia Foundation.

GOSSELIN, D. 2011. *Javascript*, Course Technology Cengage Learning.

JOSTEN, G. (2001). Chess, a living fossil. Initiative Group Königstein (IGK).

LUKE WELLING, L. T. Desarrollo Web con PHP y MySQL. ANAYA Multimedia.

MORA, S. L. 2001. *Programacion en Internet: Clientes Web*.

NORMAN, M. 2004. Database Design Manual: using MySQL for Windows. Springer.

PERDITA STEVENS, R. P. A. W. 2002. Utilización de UML en Ingeniería del Software con Objetos y Componentes.

PÉREZ, J. E. 2008. Introducción a

JavaScript.

RAUCH, G. 2011. *Socket.io* [Online]. Joyent.

SHELLY, G. B. 2009. Web Design: Introductory Concepts and Techniques. 3rd ed.: BRODKIN, A.

WIKIPEDIA. 2011. *Metodología de desarrollo de software - Wikipedia, la enciclopedia libre* [Online]. Wikimedia foundation Inc.

WIKIPEDIA. 2012a. *Aplicacion Web - Wikipedia, la enciclopedia libre* [Online]. Wikimedia Foundation Inc.

WIKIPEDIA. 2012b. *Herramienta CASE* [Online]. Wikimedia foundation Inc.

WIKIPEDIA. 2013. *Entorno de desarrollo integrado - Wikipedia, la enciclopedia libre* [Online]. Wikimedia foundation Inc.