ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería Eléctrica y Computación

"Diseño e Implementación de un Juego de Ajedrez usando TCP / IP y la Programación Cliente-Servidor"

Proyecto de Tópico de Graduación

Previo a la obtención del Título de

INGENIERO EN COMPUTACION

PRESENTADO POR:

Wilson M. Boas Arévalo

Pablo A. Tapia Bastidas

Guayaquil - Ecuador

-:- 1 9 9 5 -:-

A G R A D E C I M I E N T O

Al ING. GUIDO CAICEDO
Director del Tópico de Graduación, por su ayuda y colaboración para la realización de este trabajo.

D E D I C A T O R I A

A NUESTROS PADRES

A NUESTROS HERMANOS

DECLARACIÓN EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestos en este proyecto, me corresponden exclusivamente; y, el patrimonio intelectual del mismo, a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento de Exámenes y Títulos profesionales de la ESPOL).

.....
Wilson Manuel Boas Arévalo

.....
Pablo Anibal Tapia Bastidas

INDICE GENERAL

CAPÍTULO 1

ESPECIFICACIONES

1.1 DESCRIPCIÓN GENERAL DEL PROYECTO

El proyecto Ajedrez ESPOL, pretende implementar el juego de Ajedrez utilizando las facilidades de TCP/IP y la programación cliente servidor.

La idea central es que por medio del servicio ajedrez, un jugador (usuario del servicio) pueda establecer una partida de ajedrez con otro jugador localizado en cualquier lugar del mundo donde existan los servicios de INTERNET.

Se pretende que el ambiente en el que se desenvuelva el jugador sea gráfico, de alta calidad, amigable y que esté disponible para la mayoría de usuarios. Por los motivos mencionados se ha seleccionado a Windows 3.1 como la plataforma básica para el cliente.

Cada usuario de la aplicación podrá mantener una partida con cada uno de los otros jugadores, pudiendo seleccionar sin ambigüedad y de manera independiente cada una de las partidas que tiene pendientes y que desee seguir jugando.

Por otro lado el servicio ofrecido al usuario es NO INTRUSIVO: es decir que cada una de las jugadas o

solicitudes de los oponentes no interrumpirán las tareas que se estén realizando y es NO EN LÍNEA: es decir que las solicitudes o jugadas realizadas al oponente o al administrador no tendrán una respuesta inmediata, sino que podrán ser analizadas y contestadas en el momento que el usuario lo estime conveniente.

Las nuevas partidas pueden establecerse enviando una solicitud a cualquiera de los usuarios del sistema (oponentes), en cuyo caso la decisión de iniciar o no la partida y escoger el color de las piezas con que juega cada uno de los jugadores está en manos del oponente.

En el transcurso de una partida, cada jugada debe realizarse por medio del ratón.

Otras funciones de la aplicación cliente serán:

- Permitir que nuevos jugadores se subscriban.
- Permitir que el usuario visualice el tablero y el color de las piezas de acuerdo a sus preferencias.
- Soportar las funciones normales del juego de ajedrez, algunas de las cuales son: solicitud de tablas, promoción de peones, solicitud de ranking, aceptar derrota, etc.
- Una función interesante que deberá soportar la aplicación es la que denominaremos "SÍGUEME"; por medio de la cual el usuario no está atado a una

máquina, sino que él pudiera ir a otra máquina y continuar jugando sin problema.

En lo que se refiere a la aplicación servidora, ésta tendrá que proveer el tiempo de respuesta adecuado a todos los clientes que lo estén accesando en ese momento y su código deberá ser portable, entendible, mantenible y que además incluya código para la manipulación, autenticación y verificación de la identidad del usuario.

Además en el servidor se implementará un programa administrativo que permita realizar actividades como: ingresar nuevos usuarios, inicializar archivos, examinar solicitudes de subscripción, etc.

Cabe anotar que existe la versión 1.0 de Ajedrez ESPOL, ésta aplicación será modificada, re-diseñada y re-implementada si las necesidades así lo ameritan. En todo caso algunos de los procedimientos y funciones de ésta aplicación podrán ser utilizados en nuestro proyecto.

1.1.1 OBJETIVOS

1. Realizar un análisis del estado actual del proyecto Ajedrez ESPOL versión 1.0.

2. En base al análisis anterior establecer las modificaciones necesarias que permitan mejorar el rendimiento, confiabilidad y seguridad del programa Ajedrez ESPOL 1.0

3. Implementar las modificaciones propuestas

1.2 REQUERIMIENTOS FUNCIONALES

USUARIOS

- Un usuario puede tener una partida pendiente con cada uno de los otros usuarios.
- Un usuario debe tener un identificador único y un password que le permitan tener acceso de forma privada.
- Nuevos usuarios podrán solicitar el servicio de Ajedrez
- Si un usuario decide dejar de recibir el servicio podrá hacerlo enviando una solicitud de desubscripción.

PARTIDAS

- El sistema permitirá registrar y mantener partidas entre usuarios.
- Las partidas tendrán un nombre que las identifique de manera exclusiva de las demás.
- El esquema de codificación del nombre de partidas debe seleccionarse de modo que no cree limitaciones al sistema, permitiendo un rango suficiente de nombres de partidas para cubrir las necesidades de los usuarios.

SERVICIO

- El servicio será obtenido solo cuando el usuario lo requiera y no interrumpirá sus actividades de rutina.
- El administrador de la aplicación decidirá que nuevas solicitudes de subscripción serán aceptadas y a quien se le revocará la autorización que le permite seguir siendo usuario de la aplicación.

VISUALIZACIÓN

- Las jugadas de una partida se pueden realizar de forma simple y elegante, utilizando todas las capacidades gráficas de Windows.
- El usuario puede seleccionar el color con los que visualiza las piezas, el tablero y el fondo de la ventana.
- Para usuarios expertos se provee la opción de suprimir las coordenadas del tablero.
- Incluir la opción de selección de visualización por omisión.
- Proveer barra de estado con información sobre:
  - ◊ el estado de la comunicación
  - ◊ el color de las piezas con las que se está jugando
  - ◊ el nombre del jugador
  - ◊ el número de jugadas

◊ el tiempo que tarda el jugador en realizar la última jugada.

COMUNICACIÓN

- Minimizar el intercambio de información entre el cliente y el servidor eliminando datos redundantes.
- El sistema puede ser utilizado en redes de área local y redes de área extendida.
- Obtener tiempos de respuesta aceptables.

VALIDACIÓN

- Minimizar las posibilidades del usuario a cometer errores implementando los esquemas de validación que sean necesarios.
- Todos los movimientos de las fichas sobre el tablero serán validados utilizando el programa GNU-CHESS, con el fin de evitar jugadas ilegales.

MOVILIDAD

- El usuario no estará atado a una sola máquina, lo que permite que las partidas pendientes se continúen jugando desde otras localidades.
- Esta movilidad no afectará la consistencia de la información

ESTADÍSTICAS

- El usuario obtendrá información sobre los mejores jugadores del sistema.

SOLICITUDES/RESPUESTAS BÁSICAS

- La aplicación debe soportar la manipulación de las solicitudes y respuestas básicas que en un juego de ajedrez "normal", un jugador envía o recibe de su oponente. Estas son: solicitar tablas, aceptar tablas, rendirse.

1.3 REQUERIMIENTOS DE RENDIMIENTO Y CONFIABILIDAD

- Se espera que el tiempo de respuesta a cada uno de los comandos ejecutados por un usuario que se encuentra localizado dentro de una misma red local sea menor a un minuto.
- El tiempo de respuesta observado a una solicitud no puede ser estimado con exactitud debido a muchos factores, tales como el crecimiento de la red; pero el diseño de ésta aplicación busca encontrar un esquema que permita lograr que el tiempo de respuesta no sea sustancialmente dependiente de el número de usuarios que están accesando al servicio.
- En teoría la aplicación puede manejar un número significativamente grande de partidas (un millón), pero esto está limitado por los recursos de almacenamiento del servidor y por su Sistema Operativo.

El mal funcionamiento de un cliente no debe afectar el servicio que la aplicación servidora provee a los clientes.

1.4 HERRAMIENTAS UTILIZADAS

- El software cliente se desarrollará en Visual C**, no solo para aprovechar las facilidades de programación y los utilitarios con que cuenta éste producto sino también para usar el módulo de validación de jugadas con que cuenta el GNU-CHESS que está desarrollado usando Visual C**.
- La aplicación cliente accesa a los protocolos y servicios de comunicación haciendo llamadas a WINSOCK.DLL, el cual forma parte del paquete de comunicaciones TRUMPET versión 2.0 revisión B.
- El Servidor se desarrollará usando lenguaje C, que está incluído en el Development System para SCO UNIX debido a la disponibilidad de éstos recursos; pero la aplicación servidora estará en capacidad de compilarse y ejecutarse en cualquier Sistema Operativo UNIX.

CAPITULO 2
DISEÑO DEL PROTOCOLO

2.1 ARQUITECTURA CLIENTE-SERVIDOR DE LA APLICACIÓN

MODELO DE LA ARQUITECTURA CLIENTE-SERVIDOR

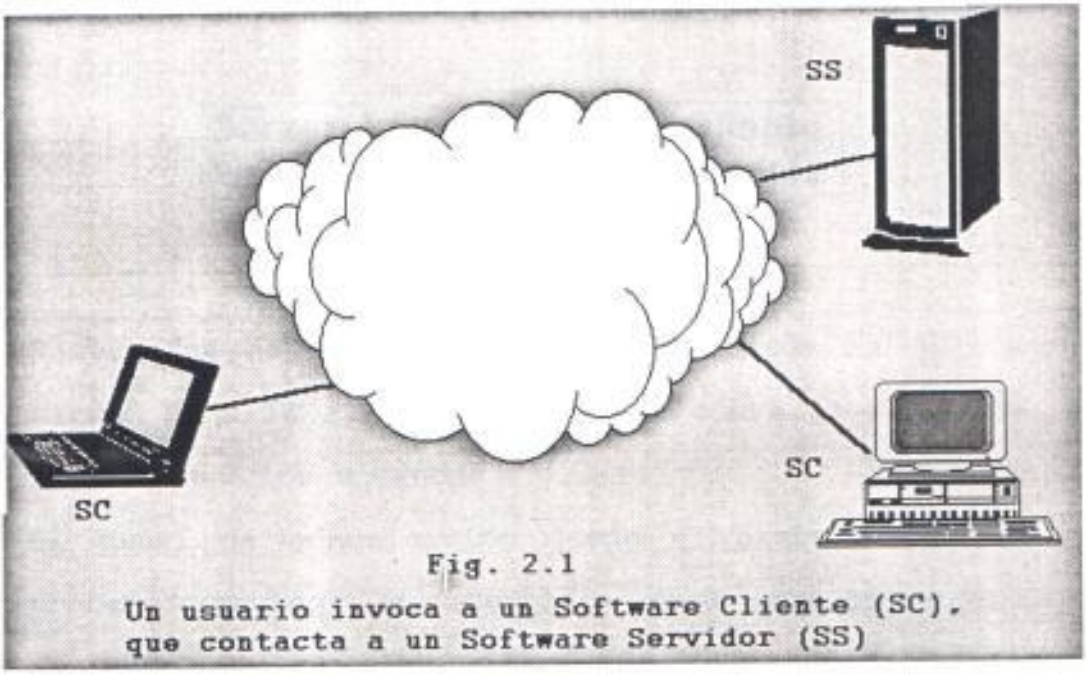


Fig. 2.1
Un usuario invoca a un Software Cliente (SC), que contacta a un Software Servidor (SS)

Cada usuario participante en el juego debe ejecutar un programa cliente y especificar el servidor al que desea comunicarse.

Luego que un cliente contacta un servidor se inicia el envío de comandos para obtener información desde el servidor; o se envían comandos para realizar solicitudes

hacia otros usuarios que puedan o no estar accesando al servidor en ese momento

El software de la aplicación utiliza un mecanismo de autenticación para asegurar que solo los usuarios autorizados puedan accesar al servidor y obtener los servicios que la aplicación AJEDREZ ESPOL v. 2.0 brinda.

Hemos definido el siguiente modelo para el servicio AJEDREZ:

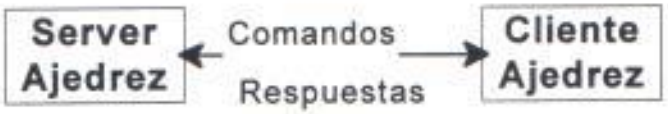


fig. # 2.2

Respuestas estándares son enviadas desde la aplicación servidora a la aplicación cliente sobre la conexión, en respuesta a los comandos.

El canal de comunicación desde el proceso cliente al proceso servidor se establece como una conexión TCP, haciendo posible que los comandos y las respuestas enviadas no sean perdidas, retrasadas o entregadas en otro orden. El canal de comunicación debe existir en todo momento durante la sesión.

Es responsabilidad del usuario cerrar la conexión cuando termine de usar el servicio ajedrez, mientras esto no se haga el servidor seguirá siendo accesado aunque no reciba comandos.

Una vez que el canal de comunicación es establecido, el cliente Ajedrez envía un comando USER indicando que desea conectarse. Si el servidor reconoce al usuario como válido envía una respuesta afirmativa al cliente, el cual envía un comando PASS cuyo parámetro es el password del usuario. Si el password es aceptado como válido, el usuario podrá accesar a todos los servicios que presta el servidor.

En base a éste modelo, en el servidor se mantiene información referente a los usuarios y a cada una de las partidas que el usuario ha establecido con sus oponentes.

2.2.- MÁQUINA DE ESTADOS

2.2.1.- MÁQUINA DE ESTADOS PARA EL CLIENTE

A continuación, en esta sección, se presentan los diagramas de estado para la implementación de Ajedrez cliente-servidor.

Existe un diagrama de estado para cada grupo de comandos de Ajedrez. Para cada comando o conjunto de comandos existen 3 posibles tipos de respuestas: éxito(S), falla (F) y error(E). El tipo de respuesta se determina analizando el código de respuesta que el servidor envía al cliente en respuesta a un comando.

Un código de respuesta de éxito (S), llega al cliente si el comando fue ejecutado satisfactoriamente en el servidor, y el cliente puede continuar con el procesamiento.

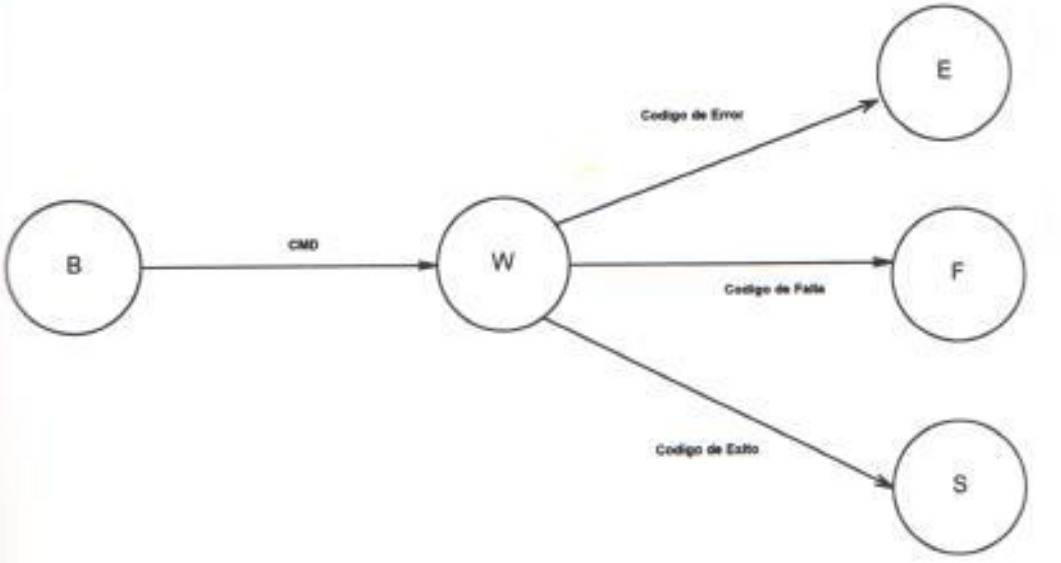
Un código de respuesta de error ocurre cuando en el servidor no se pudo ejecutar un comando debido a errores en el sistema, por ejemplo: errores al abrir, leer o escribir archivos. Por lo tanto, el cliente debe cancelar la transacción.

Los estados de falla, se originan debido a que el servidor envía un código de respuesta de falla al cliente, estos códigos de respuesta se originan cuando el servidor recibe comandos no válidos,

comandos con parámetros que no se ajustan al protocolo de comunicaciones, o comandos que no pueden ser procesados por inconsistencia en la información. En este caso el software cliente debe informar al usuario de la aplicación.

En los diagramas de estado nosotros utilizamos los símbolos (B) para indicar el inicio de la ejecución de un comando, y (W) para indicar que el cliente debe esperar por una respuesta del servidor.

2.2.1.1 DIAGRAMA DE ESTADOS PARA COMANDOS DE VERIFICACION/SOLICITUDES

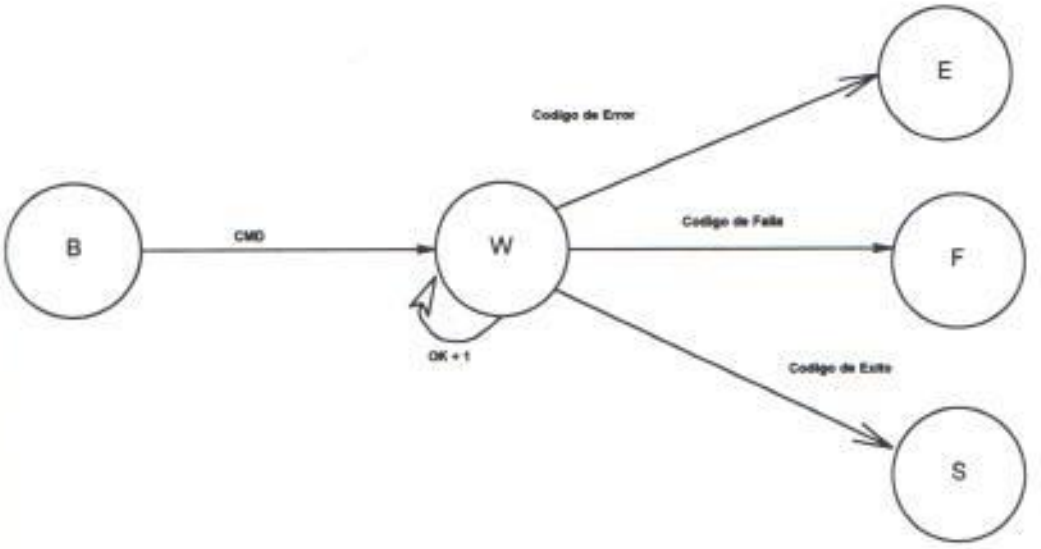


B : Inicio
E : Estado de Error
F : Estado de Falla
S : Exito
W : Estado de Espera
Cmd : Comando

Los comandos que se modelan con este diagrama de estados son: JUEG, ACEP, TBLS, RTBS, REND, SUBS, UNSU, FINP, HELP , QUIT

Fig. # 2.3

2.2.1.2 DIAGRAMAS DE ESTADOS PARA COMANDOS DE TRANSFERENCIA DE INFORMACION

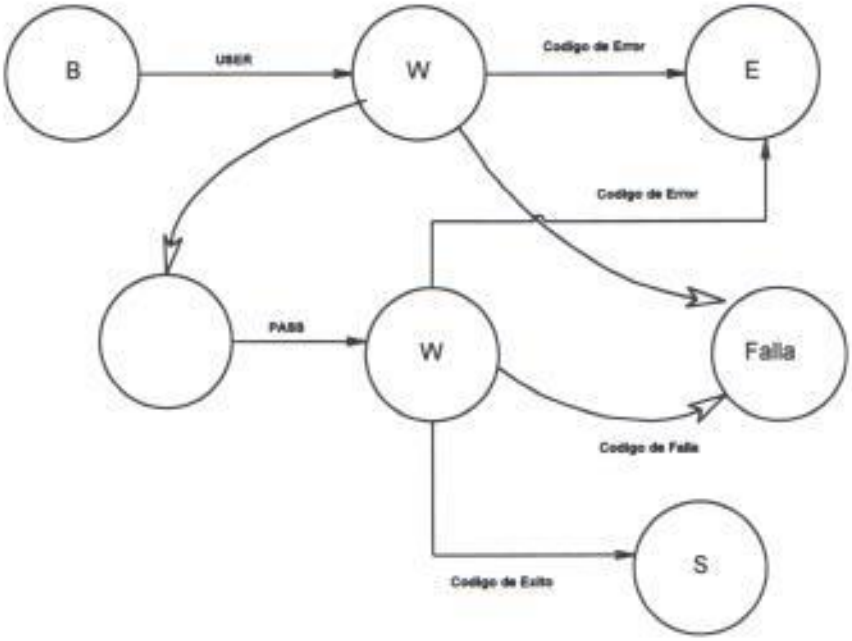


B	: Inicio
E	: Estado de Error
F	: Estado de Falla
S	: Exito
Cmd	: Comando

Los comandos que se modelan con este diagrama de estados son: JGDA, PPEN, JAUT, TRAN, SRNK.

Fig. # 2.4

2.2.1.3 DIAGRAMA DE ESTADOS PARA LA SECUENCIA DE LOGIN



B	: Inicio
E	: Estado de Error
F	: Estado de Falla
S	: Exito
W	: Estado de Espera
Cmd	: Comando

Los comandos que se modelan con este diagrama de estados son: USER, PASS.

Fig. # 2.5

2.2.1.4 DIAGRAMA DE ESTADOS PARA COMANDOS QUE SE EJECUTAN EN SECUENCIA

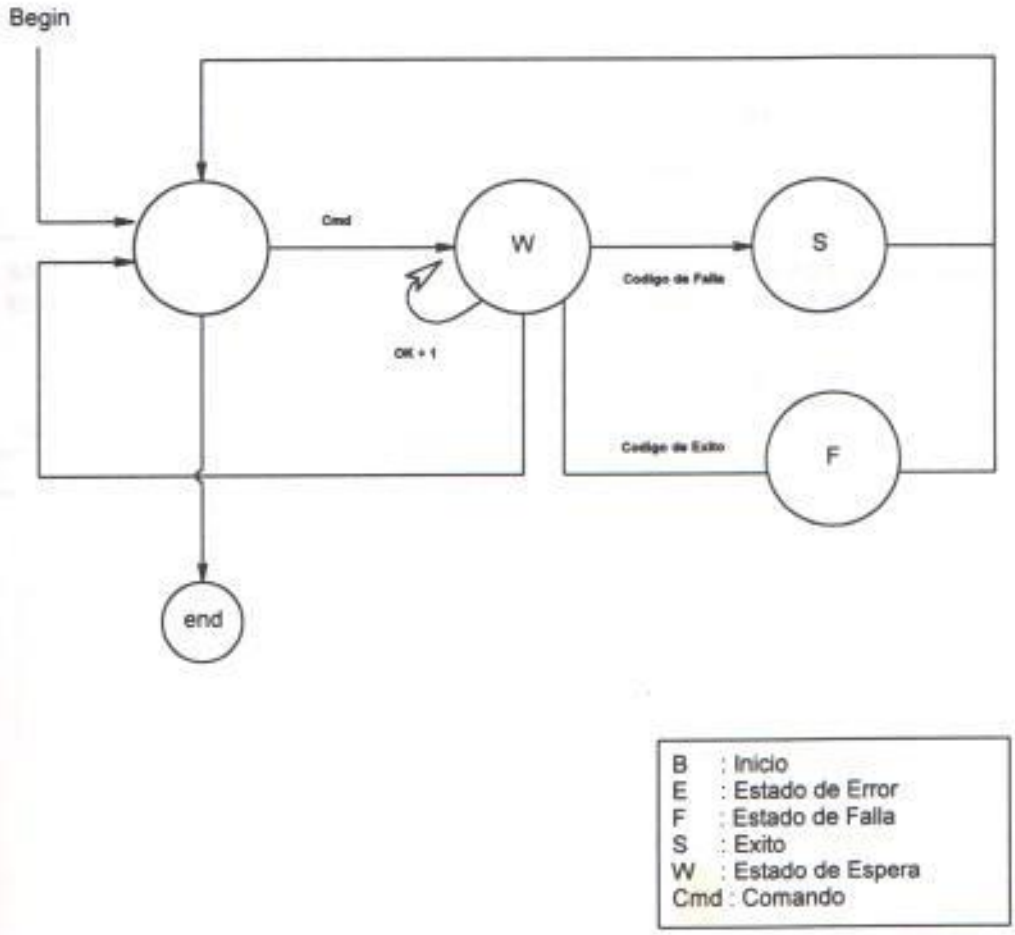


Fig. # 2.6

.2.2 MAQUINA DE ESTADOS PARA EL SERVIDOR

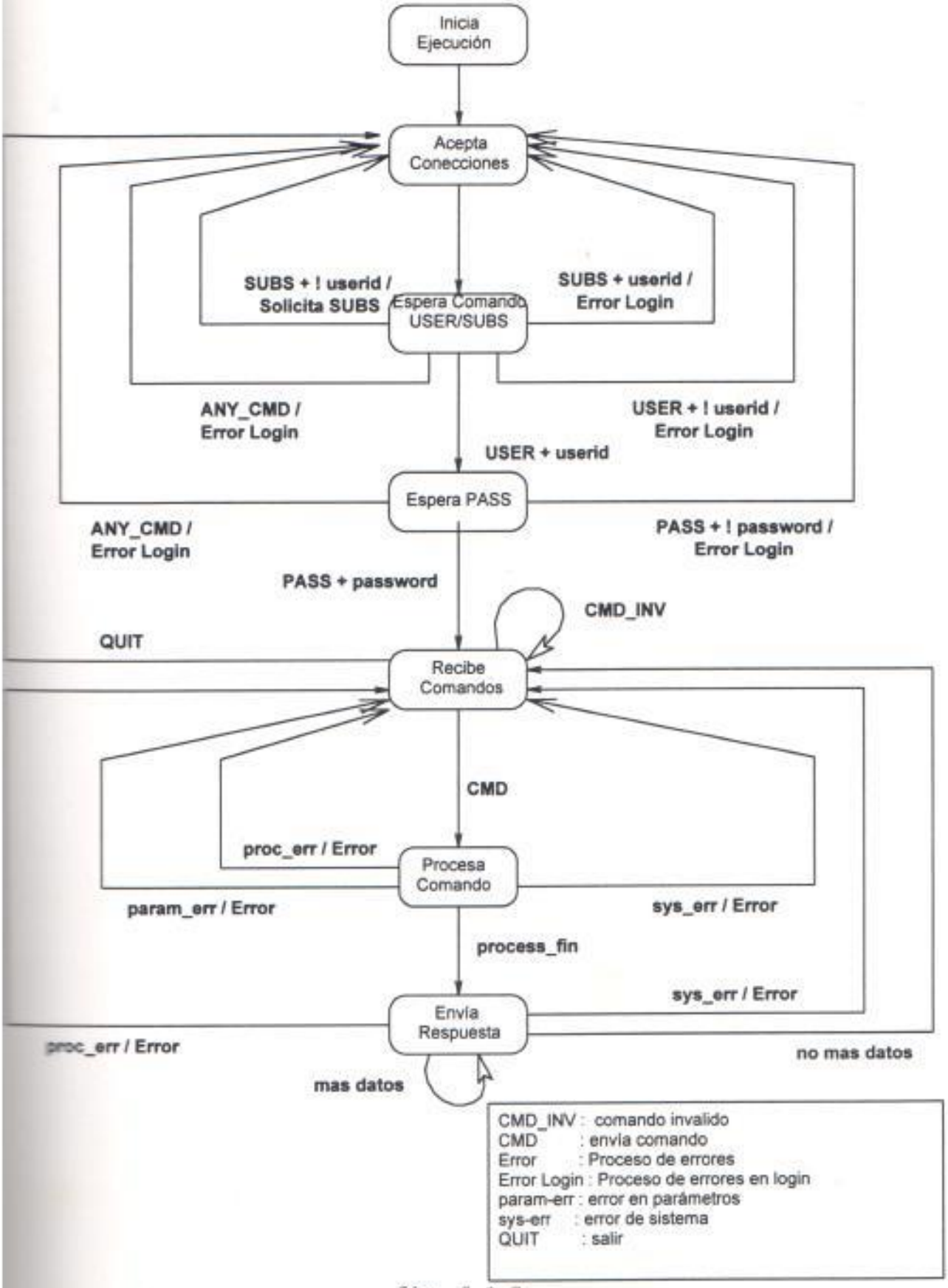


fig. # 2.7

2.3 SINTAXIS Y SEMÁNTICA DEL PROTOCOLO EN EL QUE SE BASAN EL CLIENTE Y EL SERVIDOR

2.3.1 COMANDOS :

Los comandos son cadenas de caracteres transmitidos sobre la conexión, estos comandos tienen el formato general:

< código del comando > : < argumentos > < CRLF >

El último comando en una sesión debe ser el comando QUIT.

El comando QUIT no puede ser usado en otro momento en una sesión, ya que esto involucra la terminación de la misma.

Los códigos de los comandos se forman con cuatro caracteres alfabéticos.

Estos caracteres son tratados de forma diferente si ellos representan una letra mayúscula o minúscula.

Por ejemplo: Ninguno de los ejemplos siguientes representa al comando terminar sesión (QUIT).

Quit qUit quit QuIt qUIT

Los códigos de los comandos y los argumentos son separados por ":" (dos puntos).

Un campo de los argumentos consiste de una longitud variable de caracteres. Exceptuándose los campos de tipo fecha que son cadenas numéricas de 11 caracteres de longitud.

Los campos que forman parte de los argumentos se separan entre si por medio de dos puntos.

La siguiente es una lista de los comandos para la aplicación AJEDREZ:

USER : < usuario > < CRLF >

PASS : < password > < CRLF >

TRAN : < fecha > < CRLF >

QUIT : < CRLF >

JAUT : < fecha > < CRLF >

PPEN : < fecha > < CRLF >

ACEP : < color > : < usuario destino > < CRLF >

JUEG : < usuario destino > < CRLF >

SRNK : < CRLF >

SPAR : < partida > : < #jugada > : < CRLF >

JGDA : < partida > : < #jugada > : < jugada > :
< reverso > : < posicion > : < promocion >
< CRLF >

TBLS : < partida > : < usuario destino > < CRLF >

RTBL : < partida > : < acepta tbls > :
< usuario destino > < CRLF >

REND : < partida > : < usuario destino > < CRLF >

SUBS : < usuario > : < password > : < comentario>
< CRLF >

UNSU : < usuario > : < password > < CRLF >

FINP : < partida > : < usuario destino > < CRLF >

HELP < CRLF >

STAT : < partida > < CRLF >

2.3.2 Sintaxis de los argumentos de los Comandos :

La sintaxis de los argumentos de los comandos arriba expuestos, (utilizando la notación BNF donde sea aplicable) es :

< usuario > :: = < cadena >

< password > :: = < cadena >

< usuario destino > :: = < cadena >

< comentario > :: = < cadena >

< cadena > :: = < caracter > |
< caracter > < cadena >

< caracter > :: cualquiera de los 128 caracteres
ASCII

< fecha > :: = 11 dígitos del 0 al 9

< #jugada > :: = < numero >

< numero > :: = < dígito > | < dígito > < numero>

< dígito > :: = cualquiera de los dígitos 0 al 9

< partida > :: = < código partida > < secuencial>

< código partida > :: = P

< secuencial > :: = siete dígitos del 1 al 9

< reverso > :: = 0 | 1

< promocion > :: = 0 | 2 | 3 | 4 | 5

< posicion > :: = < fuente > < destino >

< fuente > :: = 00 | 01 | 02 | ..... | 61 | 62 | 63

< destino > :: = 00 | 01 | 02 | .... | 61 | 62 | 63

< jugadas > :: = < coordenadaX > < coordenadaY >
< coordenadaX >
< coordenadaY >

< coordenadaX > :: = a | b | c | d | e | f | g | h

< coordenadaY > :: = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8

< acepta tbls > :: = S | N

< color > :: = B | N

< CRLF > :: = < CR > < LF >

< CR > :: = ascii 13

< LF > :: = ascii 10

2.3.3 SEMÁNTICA DE LOS COMANDOS DE LA APLICACIÓN AJEDREZ

COMANDO : Nombre del Usuario (USER)

SINTAXIS :

USER : < usuario > < CRLF >

DESCRIPCIÓN:

El argumento del comando es una cadena de caracteres que identifica al usuario que solicita el servicio al servidor.

RESPUESTA :

Si el comando fue procesado satisfactoriamente, el código de respuesta es 101 (S_USER), que indica que el usuario está autorizado a recibir servicio de la aplicación. Si el usuario no existe, el código de respuesta es 814 (F_NOUSER), falla por no existir usuario.

El campo de datos contiene el nombre del usuario que está recibiendo el servicio.

COMANDO : Clave de Acceso (PASS)

SINTAXIS :

PASS : < password > < CRLF >

DESCRIPCIÓN:

Envía al servidor el código de acceso del usuario al servicio.

RESPUESTA :

Si el comando fue procesado satisfactoriamente, el código de respuesta es S_PASSWD , que indica que el password es válido y permitirá tener acceso a la aplicación.

Si el password no es válido, el código de respuesta es F_PASINV que indica que el password no es válido.

El campo de datos contiene el nombre del usuario que realiza la secuencia de login.

COMANDO : Transacciones Pendientes (TRAN)

SINTAXIS :

TRAN : < fecha > < CRLF >

DESCRIPCIÓN:

Obtiene las solicitudes de juego desde la fecha especificada en el campo "fecha", que otros usuarios han enviado al usuario que ejecuta la transacción.

RESPUESTA :

Si el comando fue procesado satisfactoriamente, el código de respuesta es S_TRAN (La transacción fue ejecutada). En este caso el campo de datos contiene un buffer con todas las solicitudes de juego que han

enviado otros jugadores. En el servidor pueden existir varias solicitudes de juego, por lo tanto el formato del buffer que se envía al cliente es el siguiente:

< fecha > , < user1 > , < user2 > , .. , < usern >

< fecha > :: = fecha del sistema en el momento en el que se ejecutó la transacción.

< user > :: = usuario que realizó la solicitud

Si no existieron datos a partir de la fecha de solicitud, el código de respuesta es S_NODATA y el buffer no contiene datos válidos.

COMANDO : Terminar Sesión (QUIT)

SINTAXIS :

QUIT : < CRLF >

DESCRIPCIÓN:

Termina una sesión con el servidor.

RESPUESTA :

No necesita respuesta del servidor, una vez que el servidor recibe el comando termina la conexión.

COMANDO : Jugadores Autorizados (JAUT)

SINTAXIS :

JAUT : < fecha > < CRLF >

DESCRIPCIÓN:

Solicita al servidor la información de los jugadores disponibles para entablar un juego, y que han sido aceptados como usuarios desde la fecha descrita en el parámetro fecha. Si el campo fecha tiene el valor cero, todos los usuarios que forman parte del sistema están siendo requeridos.

RESPUESTA:

Si el comando se ejecuta satisfactoriamente en el servidor, el campo longitud mantiene la longitud de los datos enviados en el campo datos. El campo bandera llegará con el valor uno si mas datos deben ser recibidos. En el campo datos se incluyen los registros de los usuarios. Cada buffer que llega al cliente tiene el formato :

< fecha > , < user1 > , < user2 > , .. , < usern >

< fecha > :: = fecha del sistema en el momento en el que se ejecutó la transacción.

< user > :: = usuario del sistema

COMANDO : Partidas Pendientes (PPEN)

SINTAXIS :

PPEN : < fecha > < CRLF >

DESCRIPCIÓN:

Solicita al servidor la información sobre las partidas pendientes en las que un jugador participa desde la fecha descrita en el parámetro fecha. Si el campo fecha tiene el valor cero, todas las partidas pendientes para ese jugador están siendo requeridas por el cliente.

RESPUESTA :

Si el comando se ejecuta satisfactoriamente en el servidor, el campo longitud mantiene la longitud de los datos enviados en el campo datos. El campo bandera llegará con el valor uno si mas datos deben ser recibidos. En el campo datos se incluyen los registros de las partidas pendientes. Cada registro que llega al cliente tiene el formato " <partida> : <usuario> : <color> : <status> , ".

Cero o mas registros pueden llegar al cliente utilizando éste formato.

COMANDO: Aceptar Jugar (ACEP)

SINTAXIS :

ACEP : < color > : < usuario destino >

< CRLF >

DESCRIPCIÓN:

Con este comando un jugador acepta un requerimiento para jugar una partida que le ha enviado un oponente. El jugador invitado tiene la opción a escoger el color de las piezas.

RESPUESTA:

Si el comando se ejecuta satisfactoriamente en el servidor, en el campo datos se recibe el nombre del archivo que el servidor ha creado para almacenar las jugadas realizadas en la partida.

Recuerde que el nombre de este archivo tiene el formato: < partida_id > . PEN.

COMANDO: Solicitud de Juego (JUEG)

SINTAXIS:

JUEG : < usuario destino > < CRLF >

DESCRIPCIÓN:

Un jugador solicita a un oponente el establecer un juego. El oponente decidirá si desea o no establecer la partida con el usuario que envía la solicitud.

RESPUESTA:

La única respuesta que se espera es una indicación de si el comando fue o no aceptado. Si la solicitud fue aceptada el código de respuesta es S_TRAN.

Este comando falla si la solicitud ya existe o el oponente lo solicitó primero.

COMANDO : Solicitud de Ranking (SRNK)

SINTAXIS:

SRNK : < CRLF >

DESCRIPCIÓN:

Es una solicitud que un cliente hace al servidor para conocer los 10 jugadores que han ganado mas partidas.

RESPUESTA:

Buffer ordenado con los diez (10) mejores jugadores.

COMANDO : Solicitar Partida (SPAR)

SINTAXIS:

SPAR : < partida > : < #jugada > < CRLF >

DESCRIPCIÓN:

Solicita al servidor una partida, desde la jugada indicada en el numero de jugada. Si el numero de jugada es cero (0), se están solicitando todas las jugadas realizadas.

RESPUESTA:

Si el comando se ejecuta satisfactoriamente, el campo datos contiene las jugadas que se han realizado después del número de jugada especificado en la solicitud. El formato del campo datos es: "<número de jugada> : <jugada> : < reverso > : < posicion > : <promocion >". Si el número de jugada es par, la jugada la realizan las fichas negras, si el número es impar la jugada la realizan las fichas blancas. Varias jugadas pueden enviarse al cliente usando este formato. Si no se han realizado nuevas jugadas a partir de la jugada solicitada el código de retorno es F_NOMASJUG (Falla no mas jugadas). Si en la respuesta el campo bandera tiene el valor uno, el cliente debe entrar en un estado de espera para obtener las jugadas restantes.

COMANDO: Enviar Jugada (JGDA)

SINTAXIS:

JGDA : < partida > : < # jugada > :
< jugada > : < reverso > :
< posicion > : < promocion >
< CRLF >

DESCRIPCIÓN:

Es uno de los comandos mas utilizados, se utiliza para enviar una jugada y que sea recibida por el servidor y enviada al otro cliente. El numero de jugada debe incluirse para asegurar la consistencia de los datos en el servidor.

RESPUESTA:

La respuesta esperada es la indicación si el comando se ejecutó satisfactoriamente.

COMANDO: Solicitud de Tablas (TBLS)

SINTAXIS:

TBLS : < partida > : < usuario destino >

<CRLF>

DESCRIPCIÓN:

Cada partida pendiente en el sistema, tiene un registro en el archivo "est_partida" que se encuentra en el servidor. Esta transacción coloca una solicitud de tablas en el campo "solicitud_blancas" o "solicitud_negras" de este registro. El campo a actualizar depende del color con que juega el usuario que realiza la solicitud.

RESPUESTA:

La respuesta esperada es una indicación del servidor, de si el comando fue satisfactoriamente ejecutado.

COMANDO: Respuesta a Tablas (RTBL)

SINTAXIS:

RTBL : < partida > : < acepta tbls > :
< usuario destino > < CRLF >

DESCRIPCIÓN:

Es el comando que permite enviar al servidor una respuesta a la solicitud de tablas.

El campo < acepta tbls > tiene dos posibles valores que son: **(S)** para aceptar una solicitud de tablas, **(N)** para no aceptarla.

Al igual que el comando TBLS, éste comando debe actualizar el registro para la partida en el archivo "est_partida". Si el campo < acepta tbls > es **(S)** se coloca en el campo "solicitud_blancas" o "solicitud_negras" el valor prestablecido que indica que el usuario ejecutando esta transacción acepta la proposición de tablas. Si el campo < acepta tbls > es **(N)** se retira la solicitud de tablas que realizó el oponente.

RESPUESTA:

La respuesta esperada es código que indica si la transacción fue ejecutada satisfactoriamente.

COMANDO : Me Rindo (REND)

SINTAXIS:

REND : < partida > : < usuario destino >

< CRLF >

DESCRIPCIÓN:

Este comando se envía cuando uno de los jugadores decide rendirse y terminar la partida. Este comando debe actualizar el registro para la partida en el archivo "est_partida". Si el usuario ejecutando este comando juega con fichas blancas se actualiza el campo "solicitud_blancas" con el valor predeterminado que identifica que el se ha rendido. Si el usuario juega con fichas negras se actualiza el campo "solicitud_negras".

RESPUESTA:

La indicación de si el comando fue ejecutado satisfactoriamente.

DESCRIPCIÓN:

Solicita al servidor los comandos disponibles

RESPUESTA:

Buffer con todos los comandos disponibles.

COMANDO: Status (STAT)

SINTAXIS:

STAT : < partida > < CRLF >

DESCRIPCIÓN:

Solicita al servidor el status de una partida.

RESPUESTA:

Si el comando se ejecuta satisfactoriamente, el campo datos de la respuesta tiene el siguiente formato:

< partida > : < jugador_blancas > :
< jugador_negras > : < fecha > :
< status > : < ganador > :
< solicitud_blancas > :
< solicitud_negras > : < color > :
< proxima_jugada >.

2.3.4 RESPUESTAS A LOS COMANDOS

A cada uno de los comandos anteriores el servidor debe responder.

El formato de las respuestas que el servidor entrega al cliente es el siguiente.

< código de respuesta > < bandera > < longitud >

< datos > < CRLF >

2.3.4.1 Sintaxis de los campos en la respuesta

La sintaxis es la siguiente:

< código de respuesta > :: = < dígito >
< dígito >
< dígito >

< bandera > :: = < dígito >

< dígito > :: = cualquiera de los dígitos 0 al 9

< longitud > :: = entero de 16 bits,en el formato de la red

< datos > :: = < cadena >

< cadena > :: = < caracter > | < caracter > < cadena>

< caracter > :: cualquiera de los 128 carateres ASCII

2.3.4.2 Semántica de la Respuesta

Las respuestas enviadas al cliente siempre siguen un formato standard. El significado de cada uno de los campos de la respuesta es:

< código de respuesta >

Es el código que identifica el procesamiento que la aplicación servidora le ha dado a nuestra solicitud. Es un código numérico de 3 bytes. Como podemos observar este esquema nos permite mantener 1000 códigos de respuesta diferentes.
El primer dígito ha sido seleccionado para tener un significado especial.
Si la transacción fue exitosa el primer dígito será el mayor que el uno (1) y menor que el (4).
Si la transacción no fue procesada debido a un error del sistema, el código de respuesta inicia con un dígito mayor o igual que el cuatro (4) y menor que el seis (6). (ERROR)
Si la transacción detectó un error en el nombre de los comandos, en el formato de los parámetros, inconsistencias de datos , error por información por desactualización, este código inicia con un

dígito mayor o igual que el ocho (8) y menor o igual que el (9).(FALLA)

Resumiendo tendríamos:

La transacción fue procesada :

1xx , 2xx , 3xx . (ÉXITO)

Error del sistema : 4xx , 5xx . (ERROR)

Falla de procesamiento : 8xx , 9xx . (FALLA)

donde x = cualquier numero

< bandera >

Un dígito de un byte de longitud.

En la respuesta a comandos que solicitan transmisión de información si este byte esta seteado a uno (1) se le indica al cliente que debe esperar la llegada de mas datos. Si tiene el valor 0, no existen mas datos a recoger.

En los demás comandos este byte no tiene significado

< longitud >

Es la longitud de los datos que el servidor esta enviando al cliente en el bloque de datos.

< datos >

Si el código de respuesta corresponde a un rango de valores que nos indican que la transacción fue

procesada (ÉXITO), este campo contiene los datos que la aplicación transmite. Y puede ser interpretada de acuerdo a cada comando.

En el caso de ERROR o FALLA, esta contiene una cadena de caracteres que identifica el error.

NOTA : Leer anexo **A1** para una descripción de los códigos de respuesta.

2.4 JUSTIFICACIÓN DEL DISEÑO

El esquema cliente-servidor:

Algunas de las ventajas de utilizar el modelo cliente servidor en nuestro diseño e implementación se expresan a continuación:

- El modelo cliente servidor resuelve el problema conocido como "rendezvous", asegurando que cualquier par de aplicaciones se comuniquen. En este modelo un lado debe iniciar la ejecución y esperar indefinidamente que el otro se comunique.
- El modelo permitiría emigrar hacia grandes y rápidas máquinas.
- El código y los datos del servidor son mantenidos centralmente, lo cual nos permitiría realizar un mantenimiento y una administración económicos.
- Los clientes son independientes.

Por qué utilizar TCP?

Se utiliza TCP para transportar los mensajes entre el cliente y el servidor debido a que este protocolo provee la confiabilidad necesaria en un ambiente del tipo internet en el cual nuestra aplicación va a desarrollarse. Debido a la confiabilidad ganada a través

de TCP, la tarea de diseño, programación y mantenimiento se facilita.

Por qué una sola conexión maneja la interacción entre el cliente y el servidor?

Algunos protocolos de la capa de aplicación utilizan una sola conexión para intercambiar datos y comandos (TELNET), en este tipo de esquema se presenta una pequeña dificultad, cómo distinguir los datos y los comandos?. Otros protocolos utilizan más de una conexión, un ejemplo de esto es FTP, el cual establece dos conexiones, una para datos y otra para comandos, con este tipo de protocolo se puede ganar en perfomance, y en control; pero más recursos se están utilizando, además de que el protocolo se complica, requiriendo más esfuerzo en la etapa de diseño, implementación y mantenimiento.
En resumen, tener una conexión nos provee una buena perfomance, el control suficiente, y ahorros se producen en esfuerzo y recursos.

Comandos y respuestas:

El formato de los comandos:

Los comandos se envían como cadenas de caracteres separadas por un delimitador de campo, con esto logramos que los campos de cada comando puedan tener un tamaño variable. Además, terminan con una secuencia de caracteres específica **<CRLF>**, esto facilita reconocer que un comando ha finalizado.

Nos interesa tener campos de longitud variable ya que esto disminuye las limitaciones del sistema en lo que se refiere a nombres de usuarios, tamaño del password de acceso que ellos utilizan. etc.

El formato de las respuestas:

Porqué manejar una estructura igual para todas las respuestas ?

Para estandarizar, así si nuevos comandos son implementados, y siguen el estándar establecido, estos no tendrán conflicto con los ya implementados.

Los códigos de respuesta.

Para continuar con el procesamiento adecuado de la aplicación y determinar con exactitud que acción se debe tomar se utiliza el código de respuesta. Este nos dirá si

tenemos que cancelar la transacción debido a una falla del sistema, un error de formato o datos inconsistentes, o si el comando ha sido procesado adecuadamente.

El campo longitud.

Este campo permite determinar con exactitud cuántos datos debe el cliente debe esperar.

Porqué se ha incluido un byte opcional en la cabecera de una respuesta?

Existen casos en los que tal vez se tenga que transmitir bastante información; pero la aplicación tal vez no cuente en ese momento con los recursos necesarios de memoria de modo que utilizando una sola llamada se pueda transmitir toda la información; entonces este byte indicará si el cliente debe esperar por más información desde el servidor.

Secuencia de login:

La secuencia de login se utiliza para proveer la seguridad necesaria a la aplicación respecto a si las solicitudes que están procesándose corresponden a usuarios autorizados.

Además con este esquema minimizamos el envío de información a la aplicación servidora. En la versión anterior cada comando se enviaba con el usuario que requería la solicitud y el password; ahora una vez que se ha establecido la conexión, y el proceso de autenticación se ha realizado, la aplicación servidora conoce ya con exactitud de quien son las solicitudes. Esta característica no es nueva, muchos protocolos lo implementan de ésta forma, y su eficiencia y confiabilidad ya está probada.

La finalización de la conexión:

El diseño de nuestra aplicación establece que es responsabilidad del usuario cerrar la conexión, esto se debe a que la interacción entre el cliente y el servidor es mas compleja que la de enviar un solo comando y recibir una respuesta. El server en realidad desconoce que y cuántas solicitudes va a realizar el cliente. Por lo tanto el cliente debe enviar una señal de culminación para que el server conozca cuando terminar la conexión. Aunque debemos señalar que permitir que el cliente controle la duración de la conexión es **peligroso**, debido a que esto permite al cliente controlar el uso de los recursos (sockets, conexiones, etc).

Bueno, alguien tal vez propondría: "el servidor debería recibir una sola solicitud del cliente, formular la respuesta, enviar la respuesta al cliente y luego cerrar la conexión, esto ahorraría recursos", debemos indicar que en este caso el tiempo de respuesta sería afectado (recuerde que TCP requiere un gasto al crear una conexión, al tener que realizar un "three-way handshake").

CAPÍTULO 3
DISEÑO DEL SERVIDOR

3.1 Funcionalidad del Servidor

El servidor debe soportar las siguientes funciones:

- Aceptar solicitudes de conexión
- Proveer un servicio de verificación de la identidad del cliente.
- Recibir los comandos (solicitudes de los clientes)
- Procesar los comandos y formular las respuestas en el formato del protocolo de la aplicación.
- Enviar las respuestas al cliente.
- Manipular las condiciones de error
- Mantener la información individual libre de accesos no autorizados.
- Crear nuevas partidas cuando los jugadores lo hayan acordado.
- Permitir obtener un ranking sobre los mejores jugadores
- Tener la posibilidad de generar estadísticas más detalladas.

3.2 Tipo de servidor y su justificación

El servidor a implementar es un servidor concurrente orientado a conexión.

Hemos seleccionado este tipo de servidor debido a que nos provee mejores tiempos de respuesta y permite que múltiples clientes accecen al servidor simultáneamente.

Los requerimientos de la aplicación establecen que el cliente y el servidor pueden estar en la misma red de área local o en una red de área extendida, por éste motivo seleccionamos un servicio orientado a conexión debido a que éste utiliza el protocolo TCP en la capa de transporte. Esto implica que la entrega de paquetes es confiable, logrando reducir el esfuerzo de programación, diseño y mantenimiento entonces el protocolo de la capa de aplicación que diseñamos ya no debe incluir código que asegure dicha confiabilidad.

3.3 DISEÑO DE LOS DATOS MANEJADOS EN EL SERVIDOR

Para un funcionamiento eficaz que cubra todas las expectativas y funcionalidad que los usuarios requieren, el servidor debe manipular varios archivos de texto cuyo formato detallamos a continuación:

Nombre de Archivo : usuarios

Función:

Almacena la información referente a un usuario

formato por cada línea	**User : password : comentario : fecha : #partidas ganadas**
User	el nombre del usuario en el sistema
password	código de acceso
comentario	nombre completo / comentario sobre el usuario
fecha	fecha en que el usuario ingresó al sistema o fecha en que el estatus del usuario fue modificado
#partidas ganadas	El número de partidas que un jugador ha ganado

Nombre de archivo : est_partida

Función:

Almacena información del estatus de una partida y de los jugadores que participan en ella.

formato por cada línea	**Partida_id : jugador_blancas : jugador_negras : fecha : est_prt : ganador : solicitud_blancas : solicitud_negras**
Partida_id	identificador de partida
jugador_blancas	usuario que solicita jugar con piezas blancas
jugador_negras	usuario que solicita jugar con piezas negras
fecha	fecha en que la partida fue creada o terminada
èst_prt	estatus en el que se encuentra la partida Los estatus posibles son: **0** = activa, actualmente jugándose **2** = terminada
ganador	identifica quien ganó. Los valores posibles son: **0** = inicial, nadie ha ganado **1** = ganan blancas **2** = ganan negras **3** = tablas
solicitud_blancas	indica que solicitud ha realizado el jugador con fichas blancas. Los valores posibles son: **0** = no solicitud **1** = blancas solicitan tablas **2** = blancas aceptan tablas **3** = blancas abandonan

solicitud_negras	indica que solicitud ha realizado el jugador con fichas negras. Los valores posibles son: **0** = no solicitud **1** = negras solicitan tablas **2** = negras aceptan tablas **3** = negras abandonan

Nombre de archivo : P#secuencial.PEN

Función :

Almacena las jugadas realizadas en una partida

formato por cada línea	**#jugada : jugada : reverso : posicion : promocion**
#jugada	número de jugada de seis dígitos. Inician en 1. Las jugadas impares las realizan las blancas
jugada	descripción de la jugada en notación estandard de ajedrez
reverso	información de control necesaria para manipular jugadas en GNU-CHESS. Los valores posibles son: **0** = no invertir tablero **1** = invertir el tablero al realizar la jugada
posicion	formado por cuatro dígitos. Los dos primeros identifican la posición inicial de la pieza, los dos últimos la posición final. Toman valores desde 00 al 63
promocion	indica si la jugada involucra la promoción de un peón. Los valores posibles son: **0** = no promoción **2** = cambia peón por caballo **3** = cambia peón por alfil **4** = cambia peón por torre **5** = cambia peón por dama

NOTA: el # secuencial está formado por siete dígitos del 0 al 9

Nombre de archivo : Server.eje

Función:

Determina el número de proceso con el cual el proceso maestro de la aplicación servidora se está ejecutando y el número secuencial para la generación del próximo nombre de partida.

formato	**procesid : Secuencial <lf>**
procesid	# del proceso con el que se ejecuta el proceso maestro de la aplicación servidora
secuencial	# secuencial formado por siete dígitos del 0 al 9

Nombre de archivo : Server.cfg

Función:

Guarda la información de los directorios donde se encuentran los archivos: usuarios, est_partida y todas las partidas generadas.

formato	**autorizados:nombre de directorio <lf>** **est_partida:nombre de directorio <lf>** **solicitudes:nombre de directorio <lf>**
autorizados	etiqueta. El nombre de directorio a continuación de esta etiqueta indica la localización del archivo usuarios.
solicitudes	etiqueta. El nombre de directorio a continuación de esta etiqueta indica la localización del directorio donde se crearán las nuevas partidas y archivos de solicitud de juego.
est_partida	etiqueta. El nombre de directorio a continuación de esta etiqueta indica la localización del archivo est_partida.
nombre de directorio	nombre de directorio válido

Nombre de archivo : usuario

Función:

Guarda la información de las solicitudes de juego realizadas por otros jugadores hacia el usuario.

formato	**solicitante: fecha <lf>**
solicitante	nombre de la persona que realiza la solicitud de juego
fecha	11 dígitos que identifican la fecha en que se realizó la solicitud.

Nombre de archivo : subscripcion

Función:

Guarda las solicitudes de subscripción realizadas por nuevos usuarios.

formato	**SUBS : user : password : comentario <lf>**
SUBS	etiqueta
user	usuario solicitando la subscripción
password	clave de acceso del solicitante
comentario	nombre completo o dirección de correo del solicitante

3.4 DISEÑO DE LA APLICACION SERVIDORA

3.4.1. DIAGRAMA DE ESTRUCTURA PARA EL SERVIDOR

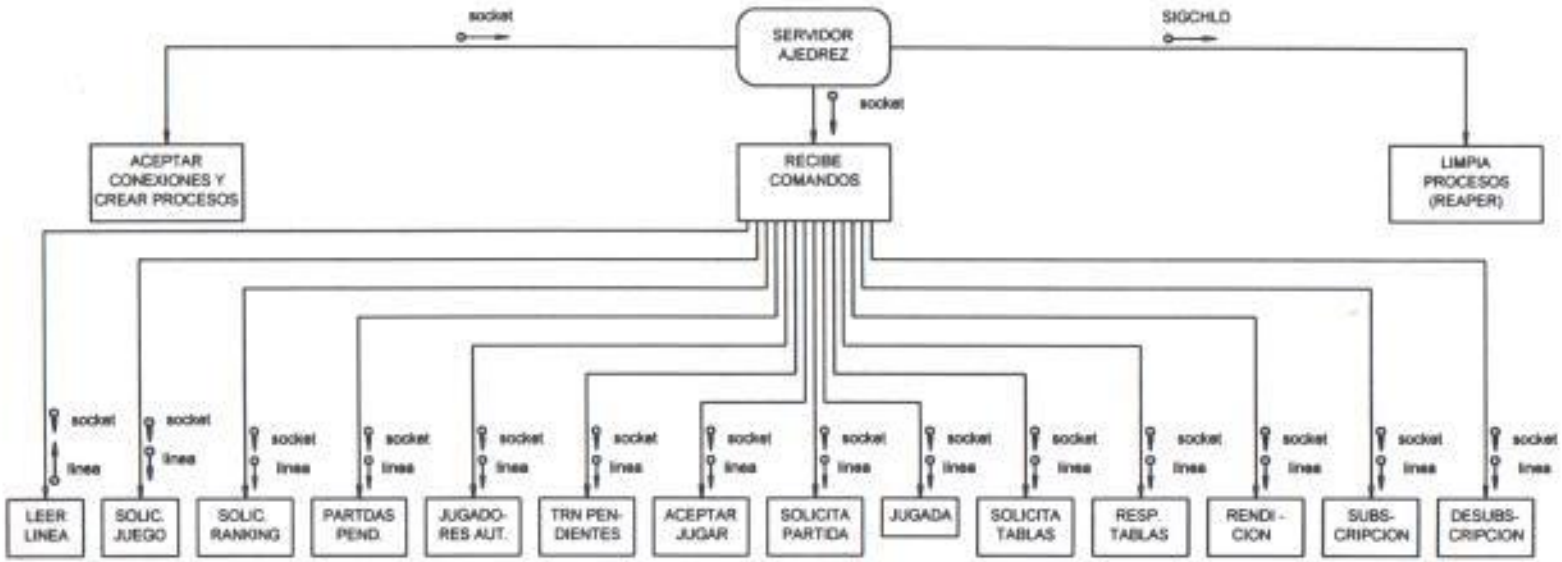


fig. # 3.0

3.4.2 ALGORITMOS PARA EL SOFTWARE SERVIDOR

Como lo anotamos en las secciones anteriores para la implementación del modelo cliente servidor que satisface nuestros requerimientos se seleccionó un servidor concurrente orientado a conexión.

En este esquema en un momento cualquiera el servidor consiste de un proceso maestro y cero o mas procesos esclavos.

El proceso maestro únicamente espera por nuevas solicitudes de conexión, cuando una solicitud arriba, el sistema retorna un socket. Este nuevo socket es el que se utiliza para comunicarse con el cliente. Después de esto el proceso maestro crea un proceso esclavo para manipular cada una de las conexiones que sean aceptadas.

El algoritmo para el proceso maestro es el siguiente:

1. Crear un socket
2. Asignar el socket a una dirección bien conocida para el servicio que está siendo ofrecido.
3. Poner el socket en modo pasivo.

Repetidamente

4. Esperar por nuevas solicitudes de conexión.

   El servidor permanece bloqueado en este punto hasta que una solicitud de conexión arribe. Cuando una

conexión llega el sistema operativo retorna un nuevo soket que sirve para manipular la conexión.

5. Crear un proceso hijo.

En el hijo:

6. Cerrar el socket maestro.
7. Ejecutar el código para recibir solicitudes y enviar respuestas (recibe_comandos).
8. Salir.

En el padre:

9. Cerrar el socket hijo.
10. Retornar al paso 4.

Como vemos este algoritmo crea los procesos esclavos para manipular cada una de las conexiones.

Una vez que una solicitud sea aceptada, el siguiente algoritmo (**recibe_comandos**) manipulará la conexión.

1. Lea una línea desde el socket.
2. procese línea
3. Si comando en línea fue USER
   Su primer parámetro identifica a un usuario?
   Si
      Lea línea
      Procese línea

conexión llega el sistema operativo retorna un nuevo soket que sirve para manipular la conexión.

5. Crear un proceso hijo.

En el hijo:

6. Cerrar el socket maestro.
7. Ejecutar el código para recibir solicitudes y enviar respuestas (recibe_comandos).
8. Salir.

En el padre:

9. Cerrar el socket hijo.
10. Retornar al paso 4.

Como vemos este algoritmo crea los procesos esclavos para manipular cada una de las conexiones.

Una vez que una solicitud sea aceptada, el siguiente algoritmo (**recibe_comandos**) manipulará la conexión.

1. Lea una línea desde el socket.
2. procese línea
3. Si comando en línea fue USER
   Su primer parámetro identifica a un usuario?
   Si
      Lea línea
      Procese línea

```
      comando en línea fue PASS?
      Si
         Su primer parámetro es el password para el
         usuario?
         Si
           Vaya a 6.
         No
           Envíe "error login"
           return.
      No
         envíe "error login"
         return.
   No
      Envíe "error login"
      return.
4. Si el comando en la línea fue SUBS
   Su primer parámetro identifica a un usuario?
      Si
        enviar al cliente "error login"
        return.
      NO
        procesar solicitud de subscripción.
        return.

5. Si es otro comando.
```

```
   enviar al cliente "error login".
   return.

6. Lea una línea desde el socket.
7. Procese línea
8. Comando es valido ?
   Si
    En caso que comando sea:
    JUEG :  solicitud_juego();
            break;
    SRNK :  solicitud_ranking();
            break;
    PPEN :  partidas_pendientes();
            break;
    JAUT :  jugadores_autorizados();
            break;
    TRAN :  transacciones_pendientes();
            break;
    ACEP :  acepta_jugar();
            break;
    SPAR :  solicitar_partida();
            break;
    JGDA :  jugada();
            break;
    TBLS :  solicita_tablas();
```

```
            break;
    RTBS :  respuesta_tablas();
            break;
    REND :  rendicion();
            break;
    SUBS :  subscripciones();
            break;
    UNSU :  desubscripciones();
            break;
    HELP :  help();
            break;
    QUIT :  return;

    default:
              enviar "error en comando"
     Vaya a 6.
   No
     envíe "error: comando inválido"
     Vaya a 6.
```

solicitud_juego:

```
/* colocar una solicitud de juego del usuario (cliente)
   en el archivo de transacciones del oponente */
1. Obtener el parámetro oponente.
```

2. Verificar que el nombre del oponente sea válido.
3. Abrir el archivo oponente para añadir.
4. Calcular la fecha actual del sistema.
5. Añadir al archivo la transacción: usuario : fecha actual.

solicitud_ranking:

/* generar el ranking para de los 10 mejores jugadores */

1. Leer el archivo usuarios
2. Ordenar el archivo usuarios de mayor a menor en base al campo partidas ganadas.
3. Obtener las 10 primeras líneas del archivo.
4. Enviar la respuesta al cliente.

partidas_pendientes:

/* generar un buffer con el formato:
" <partida> : <oponente> : <color> : <estatus > ,..."
el buffer contiene todas las partidas pendientes en las que el usuario ha participado desde la fecha que este comando trae como parámetro. */

1. Determinar el valor del parámetro fecha.
2. Leer el archivo est_partida

3. Obtener las líneas del archivo que contienen las partidas en las que el usuario participa.
4. Incluir el registro en un bufer de envío. (Repetir hasta que el buffer esté lleno o ya no existan registros en el archivo).
5. Incluir en el buffer el código de respuesta, longitud y campo opcional.
6. Enviar el buffer. Si mas datos existen borra el buffer y vaya al paso 4.

jugadores_autorizados:

/* leer el archivo de jugadores autorizados, y enviar al cliente todos los registros existentes. El formato de cada registro en el buffer a ser enviado es " < fecha > , < user1 > , < user2 > , ... , < usern > " */

1. Determinar la fecha desde la que desea el cliente realizar la consulta.
2. Seleccionar los registros cuyo campo fecha haya sido cambiado después de esa fecha.
3. Llenar datos en el buffer hasta que se llene o hasta que los datos se acaben.
4. Calcular la longitud del buffer. Incluir esta información en el campo longitud de la respuesta. Si mas datos existen incluir un uno en la cabecera. No olvidar, incluir código de respuesta.

5. Si más datos existen vaya al paso 2.

transacciones_pendientes:

/* informar sobre las transacciones pendientes al usuario* /

1. Determinar la fecha desde la que desea el cliente realizar la consulta.
2. Determinar si existe el archivo usuario. (usuario=nombre del usuario).
3. Si no existe enviar al cliente un código de respuesta S_NODATA (el comando fue ejecutado pero no existieron datos).
4. Si existe el archivo: abrir para lectura
5. Calcular la fecha del sistema y colocar en el buffer de envío
6. Leer una línea y seleccionar el registro si su campo fecha es mayor que el campo de solicitud.
7. concatenar el separador de registros (,), incluir el registro en el buffer de envío hasta que el buffer este lleno o hasta que los datos se terminen.
8. Calcular la longitud del buffer y colocar en el campo longitud. Si más datos existen incluir un uno en el campo opcional de la respuesta. Si un error ocurre o

el comando no es procesado incluir el código de respuesta.

9. Enviar el buffer.

acepta_jugar:

/* Recibe una solicitud de creación de partida, crea la partida y retorna al solicitante el código de la partida */

1. Determinar el oponente.
2. Leer del archivo server.eje el número secuencial para la creación de la próxima partida. Incrementar el número secuencial.
3. Crear el archivo P#secuencial.PEN.
4. Determinar la fecha del sistema
5. Añadir un registro al archivo est_partida con el formato: < Identificador de partida > : < jugador_blancas > : < jugador_negras > : < fecha > : 0 : 0 : 0 : 0".
6. Enviar al cliente el código de respuesta de la transacción. Incluir en el campo de datos de la respuesta el nombre del archivo creado y que deberá mantener las jugadas realizadas en la partida.

solicitar_partida:

/* solicita las jugadas de una partida desde un número de jugada. Si el número de jugada es cero todas las jugadas deben enviarse. */

1. Determinar el código de la partida.
2. Determinar el número de jugada
3. Si el estatus de la partida es terminada o la partida no existe, determinar el código de respuesta a enviar, y enviar respuesta al cliente.
4. Calcular la posición sobre el archivo en base a número de jugada.
5. Leer las jugadas siguientes. Incluir jugadas en el buffer de respuesta hasta que el buffer se llene o hasta que ya no hayan jugadas. El formato del buffer es "< #jugada > : < jugada > : < reverso > : < posicion > : < promocion > , ..."
6. Calcular la longitud del buffer. Si más jugadas existen colocar un uno en el campo opcional. Determinar el código de respuesta.
7. Enviar la respuesta al cliente. Si mas datos existen vaya al paso 6.

5. Actualizar el campo solicitud_blancas o solicitud_ negras con 1 de acuerdo a las fichas con las que juegue el usuario.
6. Enviar respuesta al cliente.

Acepta Tablas:

/* Cambia el campo de solicitud_blancas o solicitud_ negras . */

1. Determinar el identificador de partida.
2. Determinar el oponente.
3. Verificar que el usuario y el oponente sean propietarios de la partida, verificar el estatus de la partida. Si el estatus de la partida es terminada o si los usuarios no son dueños de la partida informar del error.
4. Determinar el color de fichas con las que juega el usuario.
5. Si acepta tablas es Si (S). Cambiar el campo solicitud que pertenezca al usuario con el valor 2 (aceptar tablas).
6. Si acepta tablas es No (N). Cambiar el campo solicitud del oponente a 0 (no solicitud).
7. Enviar respuesta al cliente.

Subscripción:

/* Coloca una petición de subscripción en el archivo subscripciones */

1. Determinar el nombre del usuario, el password y el Nombre completo.
2. Añadir la solicitud al archivo de subscripciones
3. Enviar código de respuesta al cliente.

Desubscripción:

/* Retira a un usuario del servicio de ajedrez */

1. Determinar el nombre del usuario.
2. Eliminar al usuario del archivo "usuarios" su estatus en el archivo usuarios.
3. Enviar solicitud al cliente.

3.5 COMPROMISOS DE DISEÑO EN EL SERVIDOR:

En la versión 1.0 del servidor Ajedrez, no existía secuencia de login; los comandos se emitian y eran acompañados del user y el password para así poder verificar si el comando era ejecutado por un usuario con su respectiva autorización. En nuestro diseño, se ha implementado una secuencia de login, en la cual la persona que desea obtener el servicio, inicialmente debe enviar su usuario y su password. Si estos son correctos, puede ingresar al servicio y enviar y recibir respuestas del servidor. Por qué se diseñó de esta forma? Pues, para no enviar información redundante en cada solicitud al servidor y entonces tratar de disminuir el tráfico en la red y la complejidad de los comandos.

Alguién podría decir que este nuevo diseño reduce la versatilidad al enviar comandos, ya que el usuario sólo podrá afectar a sus partidas y acsesar a su información; pero si en verdad pierde en versatilidad, gana en privacidad al no permitirse que alguién pueda accesar o modificar información que no le pertenece.

Por qué incluir un campo para el número de partidas ganadas en el archivo de usuarios y qué gano con esto y qué pierdo?

Un servicio que debe proveer el servidor es el brindar un ranking de jugadores. El gran dilema que se presenta es "los datos deben ser derivados (calculados cada vez) o almacenados". Debido a que el calcular cuántas partidas ha ganado un usuario, puede consumir tiempo y recursos, hemos optado por almacenar el número de partidas que cada usuario ha ganado. Ganando al mejorar el tiempo de respuesta de la solicitud, pero perdiendo espacio en el disco.

Qué se logra al incluir un campo para la fecha en cada uno de los registros de los archivos usuarios y est_partida?

Si un cliente perdiera la información local sobre las partidas pendientes de un usuario, o sobre los usuarios que pueden intervenir en nuevas partidas. No podría seguir jugando, si no existiera un esquema que le permita actualizar la información desde el servidor. Implementar este esquema traería problemas si un jugador utiliza varias máquinas. Ya que la información pudiera desactualizarse. Además problemas de perfomance se podrían presentar si no se adopta un esquema para tratar

de minimizar la información transmitida en caso de desactualizaciones. El esquema que definimos, solicita que las actualizaciones de información, en caso de ser necesarias, se realicen en función del campo fecha. Esto nos ayuda a mantener la integridad de la información, minimizar la información transmitida, y permitir que un usuario no este atado a una sóla máquina. Las desventajas que presenta es que se pierde espacio de disco, y se complica la programación.

3.6 ADMINISTRACIÓN DEL SERVIDOR

Un nuevo usuario que está solicitando el servicio de ajedrez, no puede ser arbitrariamente aceptado como tal, esta petición debe ser sometida a consideración del administrador del servidor Ajedrez. Este es uno de los motivos fundamentales por los cuales se ha pensado en crear "AdmChess". AdmChess es un pequeño programa que permitirá ingresar usuarios al sistema, inicializar el archivo server.eje y revisar las solicitudes de subscripción.

Otras posibles tareas, tales como la selección del directorio para la localización de los datos, puede realizarse en el momento de la instalación del servidor usando el archivo "server.cfg".

El puerto en el cual el servidor espera que las solicitudes de conexión lleguen no está sujeto a configuración ni a negociación.

Otras actividades administrativas tales como: crear una partida cuando los jugadores lo hayan acordado, mantener actualizado el ranking de los jugadores, deben ser realizadas de forma automática por el servidor.

CAPITULO 4

DISEÑO DEL CLIENTE

4.1 FUNCIONALIDAD DEL CLIENTE

El software cliente debe realizar las siguientes funciones:

INTERPRETACIÓN:

- Realizar la interpretación de las acciones que los usuarios realizan cuando seleccionan una opción del menú o mueven una pieza en el tablero de ajedrez, generando identificadores adecuados para la ación .
- Si los identificadores generados se refieren a solicitudes locales tales como edición de colores de fondo del tablero, selección del color en el cual las piezas del tablero se visualizan, solicitud de ayuda, modificación de preferencias de posición o coordenadas, el cliente debe realizar las acciones localmente.
- Si los identificadores se relacionan con solicitudes al servidor, el software cliente debe generar las solicitudes en el formato del protocolo de la aplicación.
- Debe interpretar las respuestas recibidas desde el servidor y generar acciones locales adecuadas a la

respuesta recibida. Si un mensaje de error se recibe, debe presentar el mensaje de error, en caso que una jugada se reciba debe generar el movimiento en el tablero.

ERRORES:

- Debe detectar errores en el canal de comunicación, en la lectura y/o escritura de archivos, en el manejo de memoria, en el ingreso de datos, etc.
- Identificar el tipo de error.
- Informar al usuario sobre el error.
- Realizar las acciones de recuperación, si fuesen posibles.

VALIDACIÓN:

- Validar que las jugadas del usuario estén de acuerdo a las reglas de ajedrez. Por ejemplo: El caballo se mueva en L, el alfil en diagonal, etc.
- Validar acciones de piezas tales como: captura de piezas, promoción de peones, enroque.
- Validar el orden en el cual cada uno de los oponentes realiza una jugada.

VISUALIZACIÓN:

- Presentar mensajes en línea que informen sobre la última jugada realizada.
- Proveer una barra de estado con información sobre: estado de la conexión, el color de las piezas con las cuales el usuario esta jugando, el nombre del jugador, el número de la jugada y el tiempo desde que el usuario ha estado conectado.
- Permitir cambiar las preferencias de visualización.

JUGADAS:

- Permitir que el usuario realice sus jugadas utilizando el ratón.
- Las jugadas del oponente se solicitan fácilmente utilizando una opción del menú.

SOLICITUDES:

- Enviar las siguientes solicitudes al servidor:
- Enviar una secuencia de login (USER y PASS),
- Solicitud de subscripción.
- Solicitud de desubscripción.
- Tablas.
- Transacciones pendientes.
- Respuesta a tablas.

- Y demás solicitudes necesarias para mantener consistente la información.

RESPUESTAS:

- Recibir las respuestas provenientes del servidor.

4.2 DISEÑO DE LOS DATOS MANEJADOS EN CLIENTE

Para un funcionamiento eficaz que cubra todas las expectativas y funcionalidad que los usuarios requieren, el servidor debe manipular varios archivos de tipo texto cuyo formato detallamos a continuación:

Nombre de archivo : USUARIOS.DAT

Función:

Almacena localmente los nombres de los usuarios autorizados para intervenir en el juego. Estos son los oponentes a los que el usuario puede enviar solicitudes para establecer partidas.

formato	**<fecha> <lf>** **<usuario> <lf>**
<fecha>	última fecha en la cual se realizó la actualización del archivo. Se coloca en la primera línea del archivo.
<usuario>	el nombre del usuario en el sistema

Nombre de archivo : usuario.prt

Función:

Almacena las partidas pendientes para el usuario que ejecuta la aplicación cliente

formato	**Partida, oponente, color, estatus <lf>**
lPartida	identificador de partida
oponente	nombre del oponente
color	color de fichas con las que juega el oponente
estatus	0 = activa 2 = terminada

Nombre de archivo : P#secuencial.PEN

Función :

Almacena las jugadas realizadas en una partida

formato	**#jugada : jugada : reverso : posicion : promocion**
#jugada	número de jugada de seis dígitos. Inician en 1. Las jugadas impares las realizan las blancas
jugada	descripción de la jugada en notación estandard de ajedrez
reverso	información de control necesaria para manipular jugadas en GNU-CHESS. Los valores posibles son: **0** = no invertir tablero **1** = invertir el tablero al realizar la jugada
posicion	formado por cuatro dígitos. Los dos primeros identifican la posición inicial de la pieza, los dos últimos la posición final. Toman valores desde 00 al 63
promocion	indica si la jugada involucra la promoción de un peón. Los valores posibles son: **0** = no promoción **2** = cambia peón por caballo **3** = cambia peón por alfil **4** = cambia peón por torre **5** = cambia peón por dama

NOTA: el # secuencial está formado por siete dígitos del 0 al 9

Nombre de archivo : usuario.trn

Función:

Guarda la información sobre las solicitudes de juego que otros usuarios le envían al usuario que ejecuta la aplicación cliente.

formato	**<fecha> <lf>** **<usuario> <lf>**
<fecha>	última fecha en la cual se realizó la actualización del archivo. Se coloca en la primera línea del archivo.
<usuario>	el nombre del usuario que realiza la solicitud de juego

4.3 DISEÑO DE LA APLICACIÓN CLIENTE

4.3.1 DIAGRAMA DE ESTRUCTURA PARA EL CLIENTE

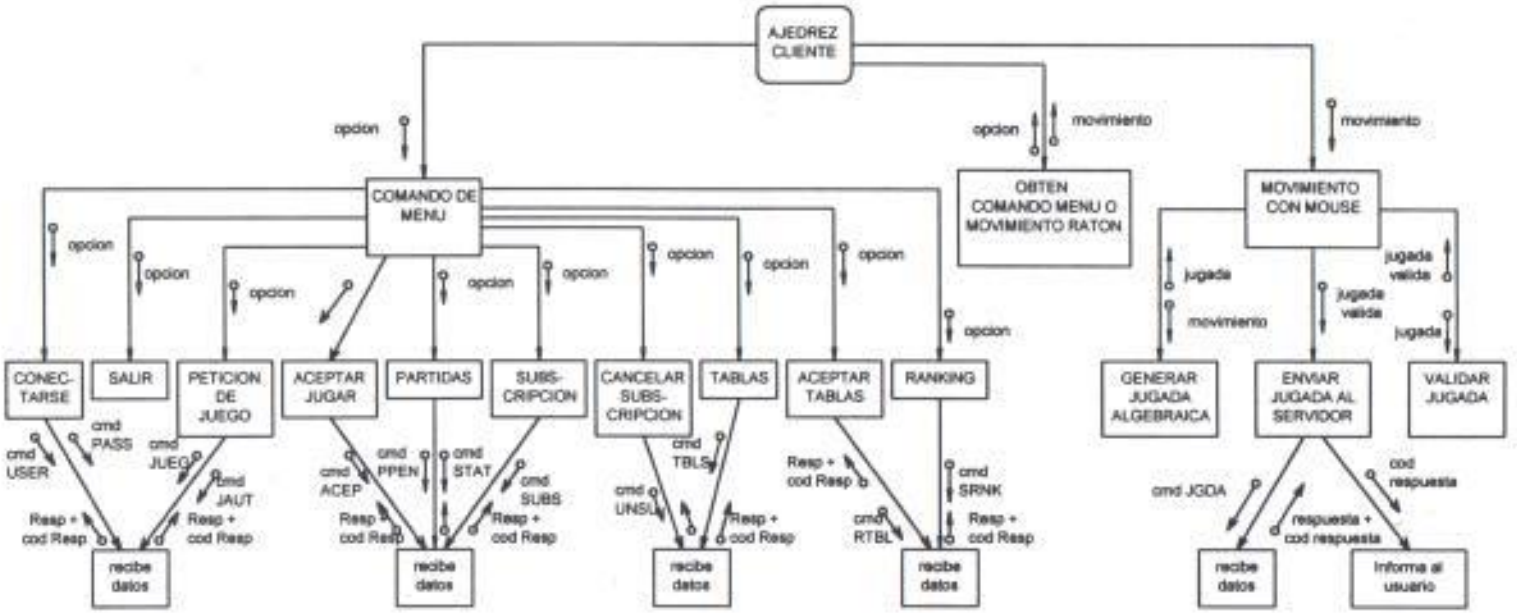


fig. # 4.0

4.3.2 ALGORITMOS PARA LA APLICACIÓN CLIENTE:

Algunos de los algoritmos para el desarrollo de la aplicación cliente son:

Petición de Juego:

/*Permite seleccionar un oponente de la lista de usuarios disponibles, y enviarle una solicitud para establecer la partida.*/

1. Obtener última fecha de actualización para el archivo USUARIOS.DAT .
2. Utilizando esta fecha, enviar una solicitud al servidor preguntando si el archivo esta actualizado.
3. El archivo esta actualizado ?

   3.1 No

   Preguntar al usuario si desea actualizar?

   No

   continuar.

   Si

   actualizar el archivo USUARIOS.DAT.

   Cancelar

   Cancela la transacción

   3.2 Si

Continuar.

4. Presentar los nombres de los posibles oponentes al usuario.
5. Permitir que el usuario seleccione un nombre.
6. Enviar la solicitud para establecer la partida.

Partidas:

/* Permite determinar la lista de partidas pendientes de el usuario, y además facilita seleccionar una para jugar */

1. Enviar una solicitud al servidor, requiriéndole las partidas pendientes que tiene el usuario.
2. Actualizar la información contenida en el archivo usuario.prt.
3. Permitir que el usuario seleccione el oponente con el cual mantiene una partida. Encontrar el identificador de la partida. El usuario puede decidir si desea continuar o cancelar.

   Continuar

      Vaya al paso 4.

   Cancelar

      Terminar transacción.

4. Determinar el color con el que juega el usuario.
5. Usando el identificador de partida solicitar el estatus de la partida al servidor. Con esta información determinar la próxima jugada a realizar.
6. Si la próxima jugada es la uno presentar tablero al usuario y terminar.
7. Si la próxima jugada es mayor que uno y no existe el archivo local de la partida, existe un error y se debe solicitar toda la partida al servidor.
   7.1 Crear el archivo para las partidas.
   7.2 Almacenar las jugadas recibidas del servidor.
   7.3 Presentar la partida al usuario y terminar.
8. Si la próxima jugada es mayor que uno y existe el archivo local para la partida.
   8.1 Calcular la próxima jugada local.
   8.2 Si la próxima jugada recibida desde el servidor es igual a la próxima jugada local. La partida está actualizada y se debe presentar la partida al usuario y terminar.
   8.3 Si la próxima jugada del servidor menos la próxima jugada local es igual a uno, estoy atrasado con una jugada por lo tanto actualizo la jugada, presento la jugada al usuario y termino.
   8.4 Si la próxima jugada del servidor menos la próxima jugada local es mayor 0 igual a dos la

partida está desactualizada. Solicito las jugadas restantes al servidor, presento la partida al usuario y termino.

Subscripción:

```
/*  Realiza los pasos necesarios para enviar una
    solicitud de subscripción al servidor */

Si no_conectado
    Establecer conexión con servidor
Presentar pantalla para recoger información
del nuevo usuario. ( el nombre de usuario con el que
desea ser identificado, el password, el nombre
completo).
Traducir la solicitud a el formato del protocolo.
Enviar la solicitud al servidor.
Cerrar conexión
```

Cancelar subscripción:

```
/*  Realiza los pasos necesarios para enviar una
    solicitud de desubscripción */
Si conectado
     Traducir la solicitud al formato del protocolo
     Enviar la solicitud al servidor
```

Historia de Juego:

/* lee un archivo de jugadas y las muestra al usuario */

1. Determinar el nombre del archivo para la partida que se esta jugando.
2. Abrir el archivo para lectura.
3. Leer las jugadas.
4. Presentar las jugadas al usuario.

Ranking:

/* Envía la solicitud de ranking al servidor, espera la respuesta y presenta la información al usuario */

1. Si está conectado
   Traducir la solicitud al formato del protocolo.
   Enviar solicitud al servidor
   Esperar Respuesta
   Si el comando fue procesado satisfactoriamente, presentar el ranking sobre los 10 mejores jugadores al usuario.

Tablas:

/* Envía una solicitud de tablas al servidor */

1. Si esta conectado :

   Determinar el identificador de la partida.

   Determinar el nombre del oponente.

   Traducir la solicitud de tablas al formato del protocolo.

   Enviar la solicitud al servidor.

   Esperar respuesta para determinar si el comando se ejecutó satisfactoriamente.

Aceptar Tablas:

/* permite aceptar solicitudes de tablas y terminar la partida */

1. Si esta conectado :

   Determinar el identificador de la partida.

   Determinar el nombre del oponente.

2. Si existe solicitud de tablas?

   2.1 Si

   Preguntar al usuario si acepta la solicitud ?

   Si || NO

   Traducir la información al formato del protocolo.

Cancelar

Cancela la transacción

Enviar la solicitud al servidor.

Esperar respuesta para determinar si el comando se ejecutó correctamente.

Rendición:

/* Informa al servidor que el jugador se ha rendido */

1. Si está conectado

Determinar el identificador de la partida

Determinar el nombre del oponente

Traducir la solicitud de rendición al formato del protocolo.

Enviar la solicitud al Servidor

Esperar la respuesta para determinar si el comando se ejecutó satisfactoriamente.

Informar al usuario.

Aceptar Jugar:

/* acepta o rechaza una solicitud de un oponente para jugar una partida, borra la solicitud de jugar partida del archivo "usuario.trn", envía solicitud al servidor */

1. Verificar que el archivo de transacciones pendientes para el usuario exista. Si no existe cancelar la transacción.
2. Mostrar los nombres de los oponentes que desean jugar partidas.
3. El jugador selecciona:

   Continuar:

      El jugador seleccionó el nombre de un jugador y el color de las fichas con las que desea jugar. Ir al paso 4.

   Cancelar :

      Cancelar la transacción.

4. Traducir la información al formato del protocolo.
6. Enviar al servidor el requerimiento.
7. Esperar por respuesta.

   El comando fue ejecutado satisfactoriamente.

   El comando tuvo error.

Salir:

/* Termina la ejecución de la aplicación */

Si está conectado

Envía requerimiento para terminar conexión

Libera memoria

Termina ejecución.

Conectarse:

Solicitar que el usuario ingrese: usuario, password y dirección del servidor

Establecer conexión con el servidor

Enviar solicitud de login: USER:usuario y PASS:password

Si no es usuario cancelar transacción.

Recibir datos del Servidor:

/* Es la función más importante y permite al cliente enviar un comando y esperar una respuesta desde el servidor. */

1. Retirar el socket del modo asincrónico.
2. Bloquear el socket.
3. Enviar el comando.

4. Recibir el código de respuesta de la transacción en el servidor. 4 bytes en formato caracter.
5. Recibir la longitud de la información que llega en el campo datos.
6. Traducir la longitud al formato del host.
7. Entrar en un lazo y esperar la llegada de la cantidad en bytes expresada en el campo longitud. Si un error en la recepción de datos ocurre, terminar.
8. Desbloquear el socket y colocarlo en modo asincrónico.
9. Terminar

4.4 COMPROMISOS DE DISEÑO:

En nuestro diseño cada uno de los clientes que contacten el Servidor de Ajedrez debe mantener información sobre: los usuarios autorizados en el sistema, las partidas pendientes que cada usuario que utiliza el software cliente está jugando con sus oponentes, las solicitudes para establecer nuevas partidas y las jugadas que el usuario ha realizado en cada partida.

Porqué mantener información en el cliente sobre los usuarios autorizados en el sistema?

Con éste esquema un jugador puede conocer y seleccionar más fácilmente los oponentes disponibles, a los cuales puede enviarles una solicitud para establecer nuevas partidas. Este archivo tendrá un nombre único: "usuarios.dat". Así, una sola copia del archivo es necesaria por cada cliente, aún si varios usuarios utilizan la misma máquina, con ésto logramos que la cantidad de información transmitida para actualizar éste archivo disminuya.

Porqué los nombres de los archivos que almacenan la información de partidas pendientes y de solicitudes para jugar nuevas partidas dependen del nombre del usuario que accesa al servicio de Ajedrez?

Para permitir que múltiples usuarios puedan compartir la misma máquina y que sus datos sean almacenados independientemente, obteniéndose un acceso más rápido a la información local de cada usuario y a la vez permite que dicha información pueda ser utilizada en el futuro.

Aunque múltiples son las ventajas de éste diseño una limitación surge debido a que DOS y Windows en la actualidad limitan el tamaño del nombre de los archivos a ocho caracteres, por lo tanto los nombres de los usuarios en nuestro sistema tendrán un tamaño menor o igual que ocho caracteres, lo que limita el número total de usuarios.

Cómo se logra reducir la cantidad de información transmitida?

Para reducir la cantidad de información transmitida la primera línea de los archivos: de usuarios autorizados, de partidas pendientes y de solicitudes para jugar nuevas partidas contienen la fecha en la que

el servidor envió datos al cliente. Por lo tanto las próximas solicitudes se realizan en base a esa fecha.

Una posible desventaja es el gasto que surge al tener que mantener actualizada ésta fecha, que es despreciable al compararlo con las ventajas obtenidas.

4.5 DISEÑO DE LA INTERFACE

Pantalla Principal

Como se observa en la figura 4.1 la pantalla principal del juego Ajedrez ESPOL v. 2.0, se basa en menús Pull-Down. Un menú es una lista de elementos o nombres que representan las acciones que puede realizar una aplicación, en nuestro caso tenemos seis menús instantáneos :

- Principal
- Partidas
- Opciones
- Preferencias
- Colores
- Help

Se han escogido los menús en vista de que son una de las herramientas más importantes que tiene Windows para la creación de interfaces sencillas y consistentes.

En todo momento, sólo se puede mostrar un menú instantáneo. El signo & hace que se muestre un caracter de subrayado bajo el caracter al que precede, lo que permite que el elemento del menú pueda ser seleccionado desde el teclado. Cuando se selecciona un menú, Windows muestra el menú en pantalla, justo debajo del elemento seleccionado en la barra de menús.

El usuario selecciona una opción con el ratón o bien desde el teclado, desplazándose hasta el elemento considerado y pulsando el botón izquierdo del ratón o pulsando una combinación de teclas.

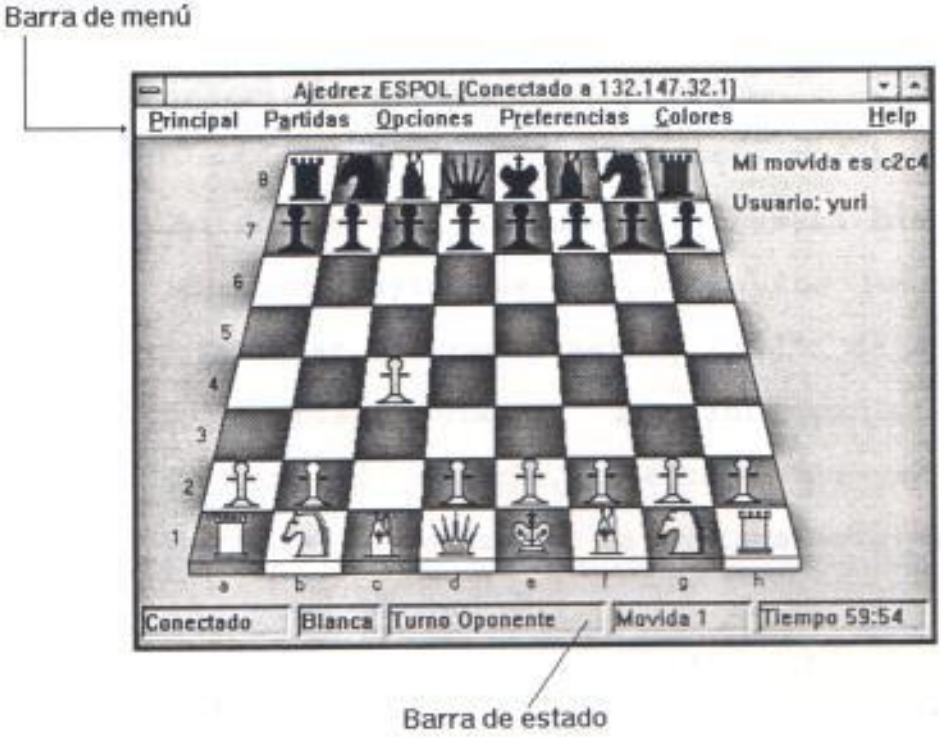


fig. # 4.1

CUADROS DE DIÁLOGO

Los cuadros de diálogo constituyen un medio más representativo para la entrada de datos. Aunque pueden introducirse datos directamente en la ventana de la aplicación, hemos preferido utilizar los cuadros de diálogo con el fin de mantener la consistencia con el resto de los programas realizados para Windows.

Los cuadros de diálogo permitirán al usuario seleccionar elementos de una lista que se presenta en una ventana, poner botones de opción o introducir cadenas desde el teclado.

Los cuadros de diálogo pueden ser construidos en dos estilos básicos: modal y sin modo. Para el desarrollo de nuestras interfaces usaremos los primeros.

Cuando se crea un **Cuadro de diálogo modal,** ninguna otra opción dentro del programa actual está disponible para el usuario, hasta que se pulsen los botones de Aceptar o Cancelar, cerrando así el cuadro de diálogo. El botón Aceptar procesa la información nueva introducida por el usuario, mientras que el botón Cancelar devuelve al usuario a la ventana original, sin procesar la nueva información.

Cuadro de Edición

Como se observa en la figura 4.2, el cuadro de edición consiste de casillas interactivas que se presentan en pantalla, para entrada de datos por parte del usuario. La información introducida, se procesará directamente.

fig. # 4.2

Botón de Radio

Este tipo de cuadro crea un pequeño círculo (el botón de radio en sí) con una etiqueta a la derecha. Los botones de radio se marcan o se verifican pulsando el botón del ratón sobre ellas, pero también pueden seleccionarse usando el teclado. Además aparecen juntos en un cuadro de diálogo y solamente se puede seleccionar un botón dentro de un grupo.

fig. # 4.3

Botón de Pulsación

Es un pequeño rectángulo de bordes redondeados y que alberga en su interior un título. Se usarán por parte del usuario para hacer una selección inmediata, tal como aceptar o cancelar el conjunto de selecciones hechas hasta el momento en el cuadro de diálogo.

fig. # 4.4

Cuadros de Selección

Este tipo de cuadro creará un recuadro con una barra de desplazamiento vertical, con el fin de permitir a los usuarios escoger un determinado elemento de varios posibles.

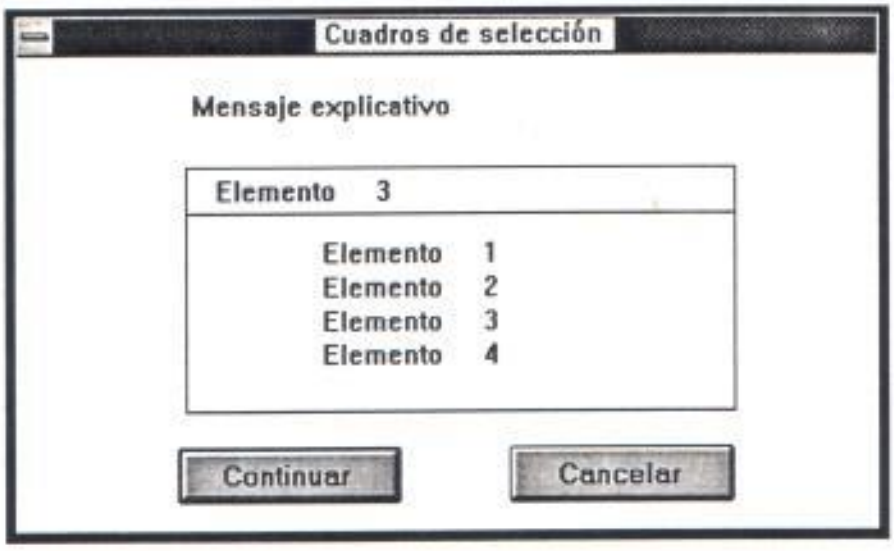


fig. # 4.5

Cuadro de diálogo para Mensaje de Error.

La siguiente ventana presenta el formato que tendrán los mensajes de error para la aplicación.

fig. # 4.6

CAPÍTULO 5

MANUALES

5.1 MANUAL DEL USUARIO APLICACIÓN CLIENTE

La interfaz gráfica del AJEDREZ ESPOL v. 2.0 hace que Ud. se adapte fácilmente a él. El uso del mouse le ayudará a aprovechar al máximo ésta interfaz.

fig. # 5.1

La pantalla de la aplicación consta de dos áreas principales: el tablero y los elementos que lo rodean a éste. En el área del tablero, se realiza básicamente el movimiento de las fichas, mientras que los elementos que lo rodean se usan para establecer: la comunicación, las preferencias del usuario, etc.

En la parte inferior de la pantalla, la **Barra de Estado** presenta información sobre el estado de su partida. Por ejemplo si observa la figura 5.1 se dará cuenta que el jugador se llama Carlos, está jugando con las fichas blancas, al oponente le corresponde hacer la siguiente jugada, el número de movida es uno y el tiempo que se demoró en realizar la jugada fué de seis segundos.

DESCRIPCIÓN DE MENÚS

Los menús se despliegan desde la barra de menús y contienen los comandos para ejecutar las tareas del Ajedrez.

PASOS PARA SELECCIONAR UN MENÚ

Haga clic en el nombre del menú o presione la tecla ALT + la letra subrayada del nombre del menú.

En la fig. 5.2 se ha seleccionado el menú principal mediante ALT + P

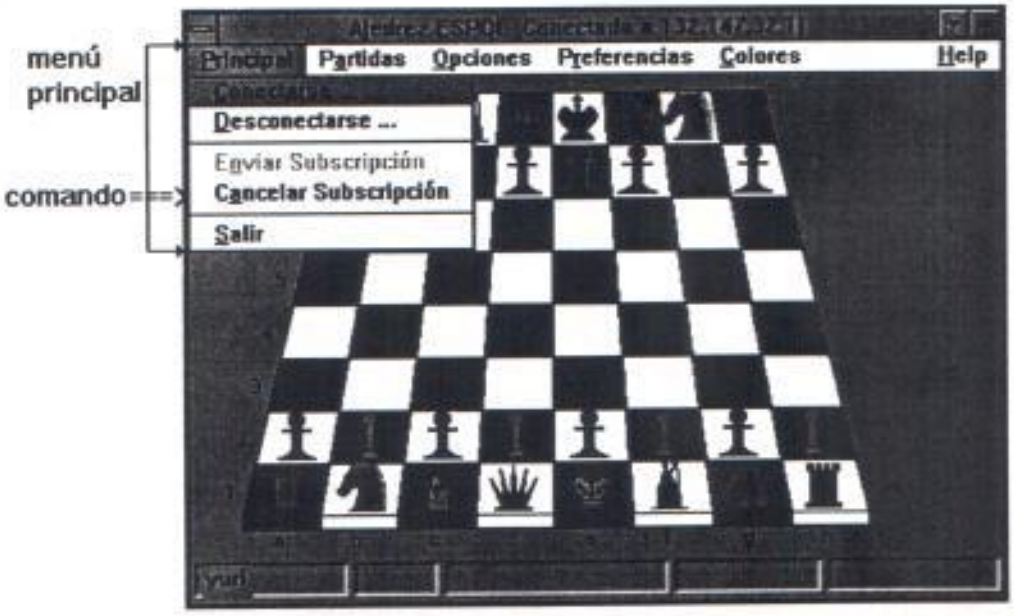


fig. # 5.2

PASOS PARA SELECCIONAR UN COMANDO

Seleccione un menú, haga clic en el comando o bien presione la tecla subrayada del nombre del comando.

Algunas de las opciones de cuadros de diálogo del Ajedrez ESPOL 2.0 son botones de comandos (ver fig. 5.3); para ejecutar un comando en estas circunstancias haga clic en el botón aceptar o presione la tecla ENTRAR con lo que la aplicación ejecuta el comando y cierra el cuadro de diálogo.

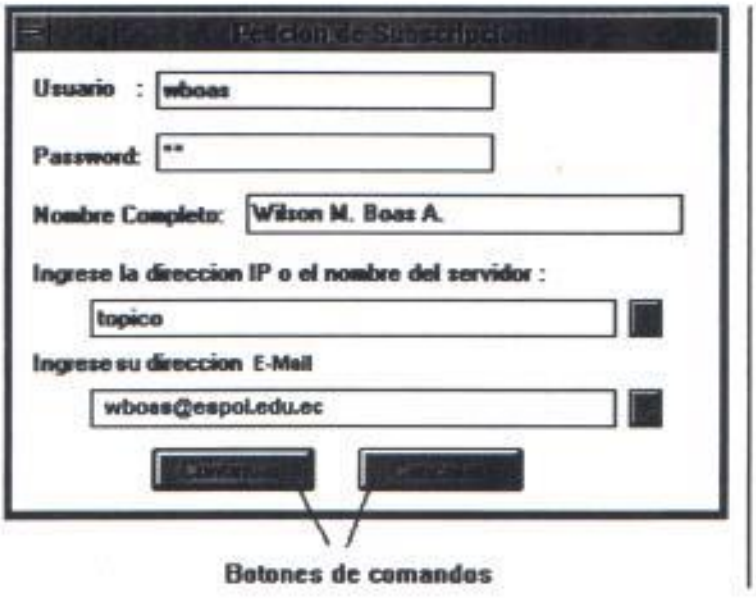


fig. 5.3

A continuación presentamos una breve descripción de los menús pull-down con que cuenta esta aplicación:

MENÚ PRINCIPAL

Conectarse	Inicia una conexión TCP con el servidor, para hacerlo hay que identificar su nombre de usuario, su password y el host con el que se desea conectar.
Desconectarse	Termina la conexión TCP con el servidor.
Enviar subscripcion	Envía la subscripción de un usuario nuevo al archivo de usuarios autorizados para jugar.
Cancelar subscripcion	Elimina a un usuario del archivo de usuarios autorizados para jugar.
Salir	Sale del juego de Ajedrez.

MENÚ PARTIDAS

Nueva partida	Acepta una petición de juego. Esto inicializa una partida y desde este momento se crea una partida y puede buscarse entre las partidas pendientes.
Partidas Pendientes	Muestra los oponentes con los cuales el usuario tine partidas pendientes.
Petición de juego	Envía requerimiento para entablar una partida.
Actualizar partida	Actualiza el tablero con la nueva jugada del oponente.
Pedir Tablas	Envia una solitud de tablas al oponente.
Aceptar tablas	Envia el mensaje al oponente de que se acepta la solicitud de tablas.
Declarar abandono	Envia el mensaje de que se abandona la partida.

MENÚ OPCIONES

Estado actual de la Partida	Presenta en pantalla información referente a la partida: nombre de los jugadores, status actual, color de las fichas.
Historia Juego	Presenta un histograma de todas las jugadas que ha hecho el usuario
Ranking Jugadores	Presenta una escala de ranking de jugadores

MENÚ PREFERENCIAS

Gira Tablero	Gira el tablero en 180 grados
Tone	Emite un tono cuando el oponente ha jugado
Coordenadas	Presenta las coordenadas del tablero
Actualizar paso a paso	Presenta los movimientos de las piezas una por una, de modo que el usuario puede tener una mejor visión de las jugadas realizadas.

MENÚ COLORES

Window Background	Define los colores para el background de la ventana
Cuadros Negros	Define los colores para los cuadros negros
Cuadros Blancos	Define los colores para los cuadros blancos
Piezas Negras	Define los colores para las fichas negras
Piezas Blancas	Define los colores para las fichas blancas
Color de Texto	Define los colores para el texto
Color Default	Define los colores por omisión

Iniciando una sesión con la Aplicación AJEDREZ ESPOL v.2.0

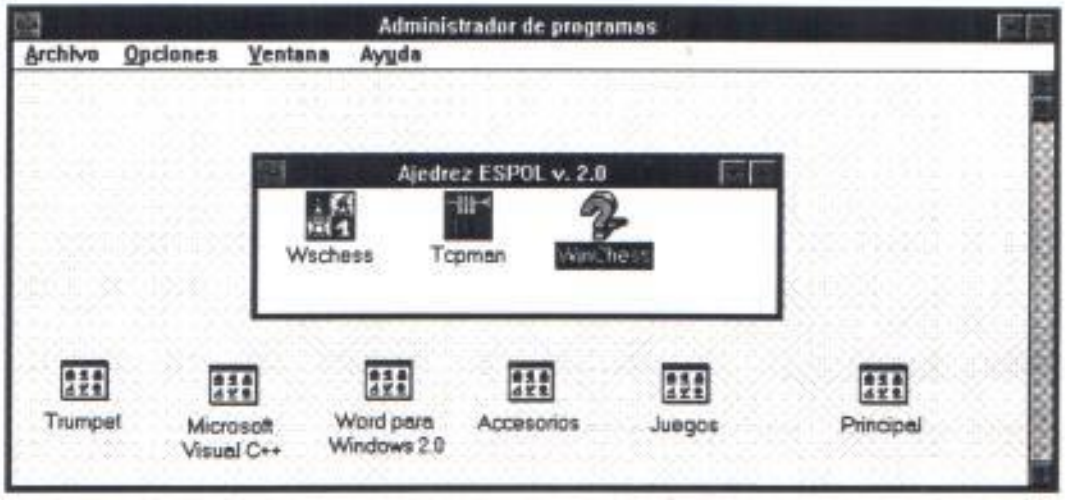


fig.# 5.4

Ejecución del Juego de Ajedrez

Primero debe ingresar a Windows y dar doble clic o ENTER sobre el ícono de la aplicación Ajedrez con esto se presenta la pantalla principal de la aplicación (fig. 5.1).

Conexión al Servidor de Ajedrez

A continuación debe seleccionar la opción *PRINCIPAL/CONECTARSE* con lo que se presenta una ventana en la que debe ingresar los siguientes datos: nombre del usuario, password y el nombre del servidor (o en su defecto la dirección IP) con el que se quiere establecer la conexión. Si la conexión tuvo éxito, se

modificará la barra de estado de la pantalla principal indicando que Ud. está conectado, caso contrario se presentará el mensaje de error respectivo.

Subscripción al Juego de Ajedrez

Si usted no cuenta con un usuario autorizado , puede enviar una solicitud de subscripción al Juego de Ajedrez, mediante el comando: ***PRINCIPAL/ENVIAR SUBSCRIPCIÓN***, con lo que se presenta un cuadro de diálogo en el que debe ingresar su nombre, password, nombre completo, nombre o dirección IP del servidor con el cual desea conectarse (ver fig. 5.3) y su dirección E-Mail.

NOTA: La ejecución de este comando no subscribe automáticamente al usuario, ya que todas las solicitudes de subscripción enviadas son analizadas por el Administrador de la Aplicación Servidora, el mismo que puede aprobarlas o rechazarlas.

Cancelar subscripción

Mediante el comando: ***PRINCIPAL/CANCELAR SUBSCRIPCIÓN***, un usuario activo emite un requerimiento para cancelar la subscripción al juego con el fin de no seguir

recibiendo datos del servidor. Esto se hace mediante una pantalla en la que se ingresa el nombre del jugador, luego su password y finalmente se pide la confirmación de la transacción.

fig. # 5.5

Desconexión del Juego de Ajedrez

Si el usuario desea terminar la sesión con la aplicación servidora debe emitir el comando: *PRINCIPAL/DESCONECTARSE*.

Esta opción permite desconectarse del servidor con el fin de preservar los recursos del mismo.

Petición de Juego

Seleccione el comando *PARTIDAS/PETICIÓN DE JUEGO*;

fig. # 5.6

A continuación se presenta la pantalla de la figura 5.6, en la que se pregunta al usuario si desea actualizar la información sobre los usuarios, puesto que el Administrador pudo haber ingresado nuevos jugadores al sistema.

Luego se presenta una pantalla con la lista de oponentes a los que se puede enviar una petición de juego, seleccione el oponente y digite ENTER o haga clic en Aceptar con el ratón (ver fig. 5.7).

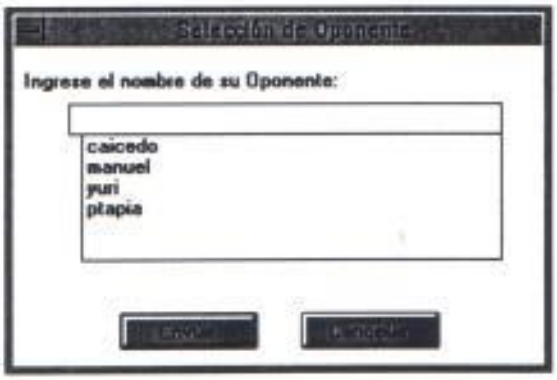


fig. # 5.7

Selección de Nuevas Partidas

Escoger del menú principal el comando: ***PARTIDAS/NUEVA PARTIDA***, luego de lo cual se presenta una ventana con el nombre de todos los jugadores que le han enviado petición de juego.

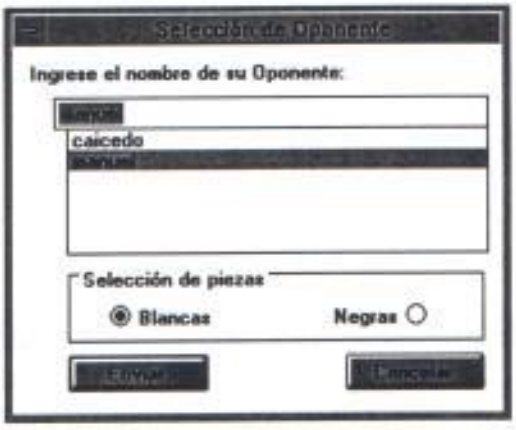


fig. # 5.8

Seleccione el jugador con el que desea establecer una partida, para ello basta con dar doble clic sobre el nombre del jugador con quien desee jugar; nótese en la

figura 5.8 que también se debe seleccionar el color de las fichas con las que desea jugar. En el caso de que no existan peticiones de juego, se presentará en pantalla la figura # 5.9.

NOTA: El jugador que acepta la invitación a jugar siempre será el que seleccione el color de las fichas.

fig. # 5.9

Selección de Partidas Pendientes

Para saber si existen partidas pendientes, basta con digitar el comando *PARTIDAS/PARTIDAS PENDIENTES*, ver fig. 5.10

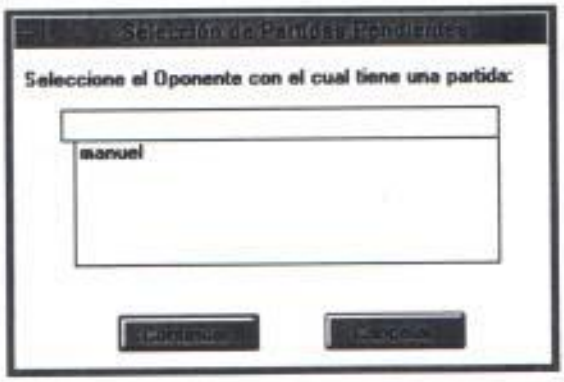


fig. # 5.10

A continuación seleccione la partida de su preferencia haciendo clic con el mouse con lo que se presentará en pantalla el tablero con las jugadas actualizadas hasta el momento en que Ud. seleccionó la partida.

Actualizar Partidas

Mediante el comando *PARTIDAS/ACTUALIZAR PARTIDA* se puede actualizar una partida con la última jugada que hizo el oponente.

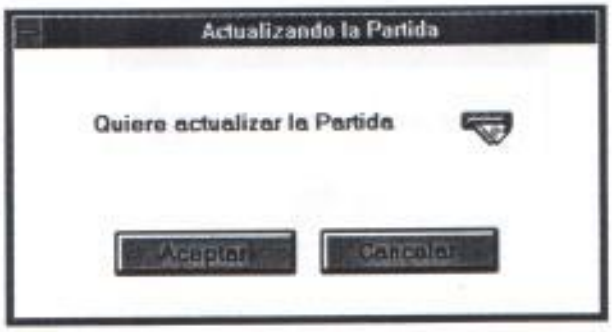


fig. # 5.11

Petición de Tablas

Mediante el comando *PARTIDAS/PEDIR TABLAS* se envía al contrincante una solicitud de tablas en la partida actual.

fig. # 5.12

Luego se presenta en pantalla un mensaje en el que se comunica que el envío ha sido exitoso.

El mensaje que se presenta al oponente cuando accese a la partida es el siguiente:

fig. # 5.13

NOTA: Cuando se envía una solicitud de tablas, el usuario no podrá realizar jugadas en la partida actual hasta que el oponente no le responda a la petición.

Aceptar/Negar Tablas

Mediante el comando *PARTIDAS/ACEPTAR|NEGAR TABLAS* se contesta al oponente a la solicitud de tablas que envió.

fig. # 5.14

Para ello simplemente se hace un clic en el botón ya sea de aceptar o no aceptar la petición de tablas (ver fig. 5.14).

NOTA: Cuando el usuario recibe una petición de tablas, la única operación que puede realizar es contestar a esa petición. Una vez que responde: si acepta la petición la partida termina en empate, caso contrario la partida continúa.

Declarar Abandono

Con el comando *PARTIDAS /DECLARAR ABANDONO* el usuario se retira del juego, lo que implica la pérdida de la partida.

fig. # 5.15

Estado Actual de la Partida

Con el comando *OPCIONES/ESTADO ACTUAL DE LA PARTIDA* el usuario puede conocer el status de la partida actual (ver fig. 5.16). Se pueden conocer datos tales como el nombre del oponente, si la partida está terminada o aún está pendiente, el color de las fichas con las que se está ejecutando.

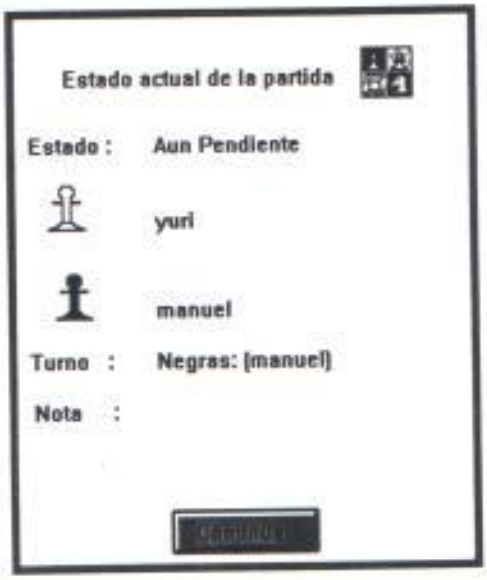


fig. # 5.16

Historia de Jugadas

El jugador, podrá en cualquier momento solicitar una historia de jugadas con el COMANDO *OPCIONES/HISTORIA DE JUGADAS* con lo que se presenta una pantalla con todas las jugadas realizadas de una partida determinada, basándose en las coordenadas del tablero.

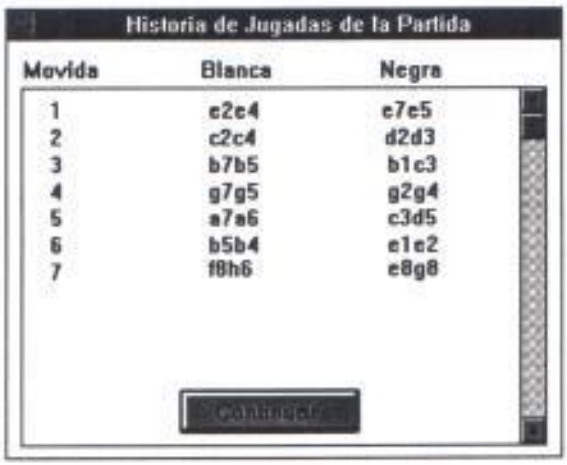


fig. # 5.17

Ranking de Jugadores

Con el comando *Opciones/Ranking* se muestran los jugadores del sistema con mayor número de partidas ganadas en orden descendente.

fig. # 5.18

Cambio de colores

Los colores de la aplicación pueden cambiarse de acuerdo a los gustos del usuario. Para ello, se selecciona el menú Colores (Alt + C) de la pantalla principal luego de lo cual se despliegan las opciones con que cuenta este menú.

Al seleccionar cada una de estas opciones, se presentará una pantalla como la de la figura ;de esta forma se pueden cambiar los colores para el fondo de los cuadros, el fondo de la pantalla, las fichas y los textos. La última opción de este menú permite seleccionar los colores por omisión con que cuenta el Ajedrez.

Opción About

El usuario puede pedir información acerca de la aplicación del juego de Ajedrez con la opción About, presentándose la pantalla de la figura 5.20.

fig. # 5.19

Promoción de Peones

Una de las reglas del juego de Ajedrez es que cuando un peón llega al extremo del tablero contrario debe realizarse la promoción del peón; para esto en la pantalla de la figura 5.21 se selecciona con que ficha se promociona el peón.

fig. # 5.20

Cómo jugar Ajedrez ESPOL 2.0 ?

NOTA: Esta sección asume que Usted está inscrito en el juego de Ajedrez. Si aún no lo está envíe una solicitud de subscripción con todos sus datos al Administrador de la aplicación Ajedrez. La respuesta a su solicitud será enviada a su dirección de correo.

En primer lugar Usted debe iniciar una sesión y conectarse al Servidor de Ajedrez.

Para establecer una partida puede hacerlo de dos formas:

1. Si es la primera vez que ingresa al juego seleccione la opción Partidas/Petición de Juego que le permite enviar a sus oponentes una solicitud para establecer una partida. Cuando el oponente acepta la solicitud que Ud. le envió, se ha establecido una nueva partida.
2. Del menú Partidas, seleccionar la opción Nueva Partida, ésto le permite revisar las solicitudes de partidas enviadas por sus oponentes y seleccionar la de su preferencia. Una vez que acepta alguna solicitud se ha establecido la partida. Ud puede establecer una sola partida con cada uno de los oponentes del sistema. Si quiere

tener más de una partida con un mismo oponente se le enviará un mensaje de error.

Cuando el usuario ha establecido partidas, las puede examinar con la opción Partidas Pendientes, la cual presenta un menú con el nombre de todos los oponentes con los que el usuario tiene partidas aún no terminadas. Seleccione el oponente de su preferencia y continúe jugando.

Movimiento de las fichas

Para mover una ficha sobre el tablero haga clic con el botón izquierdo del "ratón" sobre la ficha deseada y arrástrela hacia la nueva posición. Tenga en cuenta que todos los movimientos de las fichas están validados, por lo tanto si ha realizado un movimiento ilegal el sistema le enviará un mensaje de error.

Cuando un usuario realiza una jugada, ésta se almacena en el Servidor. Cuando el oponente responde

Para mover una ficha sobre el tablero haga clic con el botón izquierdo del "ratón" sobre la ficha deseada y arrástrela hacia la nueva posición. Tenga en cuenta que todos los movimientos de las fichas están validados, por lo tanto si ha realizado un movimiento ilegal el sistema le enviará un mensaje de error.

Jaque Mate

En Ajedrez ESPOL 2.0, no existe "Jaque Mate". Cuando en una partida dada se ha llegado a una situación de jaque mate, el perdedor tendrá que emitir el comando: Partidas/Declarar Abandono. Si no emite éste comando, cada vez que el usuario trate de mover una ficha, el sistema le enviará un mensaje de movimiento ilegal.

5.2 MANUAL DEL ADMINISTRADOR APLICACIÓN SERVIDORA

Esta guía está diseñada para describir las tareas que el Administrador debe realizar para asegurar el buen funcionamiento de la aplicación Ajedrez ESPOL v. 2.0.

Para ello, hemos diseñado una interface tipo menú en la que se harán mucho más sencillas dichas tareas.

Para ingresar al módulo de administración de la aplicación, se ejecuta el comando **AdmChess** luego de lo cual se presentará la siguiente pantalla:

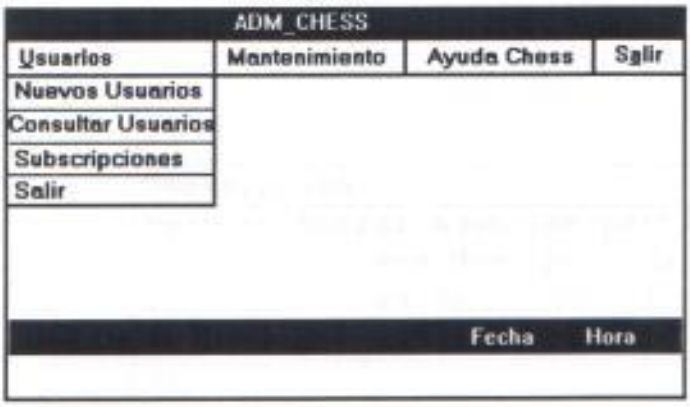


fig. # 5.21

Para seleccionar las opciones del menú, presione la tecla ALT + la letra subrayada del nombre del menú, por ejemplo si desea seleccionar el menú Usuarios se lo hace mediante la combinación ALT + U. Si desea cancelar un menú presione dos veces la tecla ESC o digite la tecla F5.

DESCRIPCIÓN DE OPCIONES

MENÚ USUARIOS

Opción	Descripción
Nuevos Usuarios	Permite crear cuentas de nuevos usuarios que deseen accesar al servicio Ajedrez
Consultar Usuarios	Permite consultar las cuentas de usuarios que pertenecen al servicio Ajedrez.
Subscripciones	Permite visualizar todas las solicitudes de subscripción de las personas que desean ingresar a la Aplicación.
Salir	Sale del menú actual.

MENÚ MANTENIMIENTO

Opción	Descripción
Eliminar Partidas	Permite borrar aquellas partidas cuyo status sea dos (partida terminada), es decir que ya no se están jugando.
Salir	Sale ál menú principal.

MENÚ AYUDA CHESS

Opción	Descripción
Ayuda Chess	Invoca al módulo de ayuda que posee la aplicación administradora.

MENÚ SALIR

Opción	Descripción
Salir	Sale del módulo de Administración de la aplicación servidora

CREACIÓN DE USUARIOS

Para crear nuevas cuentas de usuarios, el Administrador debe seleccionar el comando *USUARIOS/NUEVOS USUARIOS*, luego de lo cual se presentará una ventana similar a la de la figura 5.18

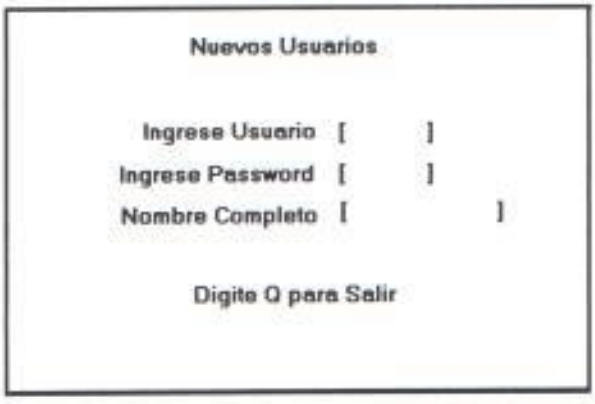


fig. 5.22

En el campo **INGRESE USUARIO**, se colocará el Userid con el que se identificará al usuario en la aplicación. Dicho campo es una cadena de caracteres alfanumérica de hasta ocho caracteres, por ejemplo creinoso, wboas, pchavez1, describen nombres de usuarios.

En el campo **Password** se colocará la clave del usuario, que de igual manera es una cadena alfanumérica de hasta ocho caracteres. Finalmente se pide el nombre completo del usuario. Este campo es un comentario de modo que facilite la identificación del usuario.

RETIRAR USUARIOS

Para retirar usuarios de la aplicación, lo hacemos en la opción de CONSULTAR USUARIOS, puesto que se invoca al editor de texto del UNIX (por ejemplo el VI). Esto permite eliminar mediante un simple comando toda la línea que identifica al usuario.

Mediante éste método se pueden cambiar tanto el password como el Nombre Completo del usuario, es decir que en el editor de texto reemplazamos los valores anteriores por los actuales.

OTRAS TAREAS ADMINISTRATIVAS

La selección del directorio para la ubicación de los datos se realiza en el momento de la instalación del servidor usando el archivo **server.cfg**, que guarda información respecto a la localización de los archivos : usr_autorizados, est_partida y todas las partidas generadas.

5.3 MANUAL DE INSTALACIÓN

5.3.1 APLICACIÓN CLIENTE

Para instalar la aplicación cliente Ajedrez Espol con Windows Sockets usted debe realizar los siguientes pasos:

1. Verifique que TRUMPET WINSOCK está correctamente instalado en su sistema. El Trumpet Winsock es una pila de protocolos TCP/IP compatible con Windows Sockets 1.1 que le provee las capas de red (networking layer) estándar para la aplicación. Un buen documento para realizar la instalación de TRUMPET es INSTALL.WRI, el cual acompaña a dicho producto.

2. Recuerde modificar el path en su archivo autoexec.bat para que contenga una referencia al directorio donde instalo TRUMPET.
   Ejemplo. path c:\dos;c:\windows;c:\trumpet
   Asegúrese que está activo rebooteando la computadora o ejecutando su autoexec.bat otra vez.

3. Antes de continuar copie los archivos wschess.exe y wschess.hlp a un directorio conveniente.

   Por ejemplo: c:\Ajedrez

   Ahora está listo para iniciar Windows. Inícielo y Disfrute Ajedrez Espol con Windows Sockets.

5.3.2 INSTALANDO EL SERVIDOR

El archivo ejecutable para el servidor se denomina chess. Si este archivo no existe pero usted cuenta con la versión en código fuente del software servidor, utilice un makefile que está incluido en el paquete. Recuerde solo debe ejecutar la instrucción "make".

Verifique que exista el archivo server.eje. Si éste archivo no existe, las nuevas partidas no podrán ser generadas. Si está realizando la instalación por primera vez, inicialice el segundo campo de la primera línea de este archivo. Coloque siete dígitos, todos ceros. Así la primera partida que los usuarios generen tendrá el nombre P0000000.PEN. Recuerde que el archivo server.eje tiene solo una línea con dos campos, el separador de campos es el ":" dos puntos. Para modificar el archivo utilice un editor de texto como el "vi".

El próximo paso que debe realizar es modificar el archivo server.cfg. Este archivo debe contener el nombre de los directorios donde se encuentran los

archivos: usuarios, est_partida, y el nombre de los archivos donde se generaran las partidas y las solicitudes de subscripción.

En la primera línea de este archivo existe una etiqueta llamada "autorizados" luego de la cual existe el nombre de un directorio, a continuación de esta etiqueta coloque el nombre del directorio donde desea almacenar el archivo de usuarios del sistema.

En la segunda línea existe una etiqueta llamada solicitudes, luego de los dos puntos que siguen a ésta etiqueta coloque el nombre de de un directorio válido donde usted desea que se generen el archivo de solicitudes de subscripcion.

La última línea de este archivo contiene la etiqueta est_partida , luego de esta etiqueta coloque el nombre del directorio donde desea que se creen las nuevas partidas y el archivo de registro de partidas est_partida.

Una vez que ha configurado este archivo, usted está listo para ejecutar el programa servidor.

Le deseamos éxito.

CONCLUSIONES

1. Utilizar un servidor orientado a conexión es sinónimo de usar TCP por lo tanto la confiabilidad de la comunicación de nuestra aplicación está garantizada.

2. El protocolo de la aplicación Ajedrez es flexible. Esto implica qure ni el cliente ni el servidor necesitan entender la Arquitectura del otro, facilitando la migración hacia otras plataformas.

3. Aunque el desarrollo de protocolos de aplicación orientados a caracter, facilita las etapas de implementación , pruebas y mantenimiento, mejoras sustanciales se pueden producir al transmitir información numérica (enteros) entre el cliente y el servidor utilizando el Network Byte Order.

4. No se ha realizado un análisis matemático profundo sobre la disminución de la cantidad de información transmitida entre el cliente y el servidor, debido a que un análisis de esta naturaleza cubre temas fuera del alcance de este proyecto; pero la lógica y los cálculos fundamentales sugieren que una disminución de la información transmitida definitivamente ocurre.

5. INTERNET es una gran fuente de conocimientos, la que debe ser aprovechada por los investigadores ecuatorianos, y sin la cual este proyecto no hubiese podido realizarse.

BIBLIOGRAFÍA

COMER, D. E. [1991], Internetworking With TCP/IP Vol I: Principles, Protocols and Architecture, 2nd edition, Prentice-Hall, Englewood Cliffs, New Jersey.

COMER, D. E. [1993], Internetworking With TCP/IP Vol III: Client-Server Programming and Applications BSD Socket Version, Prentice-Hall, Englewood Cliffs, New Jersey.

FAIRLEY, Richard [1987], Ingeniería de Software, McGraw-Hill, México D. F., México.

MURRAY, William H. III y PAPPAS, Chris H. [S/F], Programación en Windows 3.1, Osborne/McGraw-Hill, Madrid, España.

POSTEL, J. B. [1982], RFC821: Simple Mail Transfer Protocol, Information Sciences Institute, University of Southern California, California, USA.

POSTEL, J. B. and REYNOLDS, J. [1985], RFC959: File Transfer Protocol, Information Sciences Institute, University of Southern California, California, USA.

SMITH, Patrick and BSG [1992], Client/Server Computing, SAMS, Indiana, USA.

SOMMERVILLE, Ian [1988], Ingeniería de Software, Addison-Wesley Iberoamericana, México D. F., México.

STEVENS, W. R. [1990], UNIX Network Programing, Prentice-Hall, Englewood Cliffs, New Jersey.