ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería Eléctrica y Computación

PROYECTO DE GRADUACION

"*Administrador de Agenda*"

Previa a la obtención del Título de

INGENIERO EN COMPUTACION

PRESENTADA POR:

Jonathan Chávez G.

Guillermo Franco S.

Wilson Narea S.

Wilson Reinoso J.

Guayaquil - Ecuador

-:- 1 9 9 5 -:-

INDICE

CAPITULO I

1. ESPECIFICACIONES.

1.1. DESCRIPCION GENERAL DEL PROYECTO.

El siguiente proyecto es un manejador de agenda, el cual permite a un usuario planificar sus actividades diarias. Este tipo de agenda puede ser unipersonal o compartida, permitiendo a más de un usuario, además del propietario de la agenda, poder ver las actividades de éste, o si tiene atribuciones para hacerlo, poder ingresar actividades adicionales a las ya existentes.

Esta agenda se basa en el esquema Cliente - Servidor, donde el programa servidor que está en una computadora con sistema operativo **UNIX,** interactúa con el programa cliente, el cual se encuentra en un PC y que funciona bajo **Windows**. La conexión entre estas dos máquinas se logra gracias al protocolo de transporte **TCP/IP** que debe ser soportada por ambos sistemas.

La función principal del servidor es la de manejar los archivos de las agendas, para poder agregar, modificar, eliminar o consultar actividades en ellas, así como también , manejar además el archivo de usuarios autorizados al servicio. El programa cliente se encarga por otra parte de permitir al usuario ingresar sus datos haciendo uso de una interfase amigable y eficiente, esto se hace explotando las facilidades gráficas que provee Windows, tales como botones, cajas de edición, menus tipo pop-down y de vantana, etc, además del uso del mouse, por supuesto; mecanismos que permiten al usuario una interacción rápida, fácil y eficiente con el programa.

El programa cliente debe además encargarse de verificar que el usuario tenga acceso al servicio de la agenda, después de esto, le debe permitir ejecutar las operaciones que considere conveniente y también darle prioridad al dueño de la agenda para permitirle a otros usuarios ver y talvez añadir actividades a su agenda.

1.2. REQUERIMIENTOS FUNCIONALES.

- Manejar los archivos en donde se contienen las agendas de los usuarios autorizados al servicio, para que de éstos puedan añadir, modificar o eliminar datos de sus respectivas agendas.
- Debe permitir a un usuario (si este está autorizado), a añadir actividades en la agenda de otro usuario, los datos que este usuario añada no deben ser alterados por otro usuario, que también tenga acceso a la misma agenda.
- Permitir al administrador de la agenda añadir o eliminar nuevos usuarios autorizados al servicio de la agenda.
- Autenticar el uso del servicio, de forma que un usuario que no tenga acceso al mismo no pueda hacer uso de éste.
- Debe permitir ingresar, modificar y eliminar las actividades que cada usuario autorizado al servicio desee ingresar.
- Permitir al usuario tener un login name y un password, para de esta forma poder determinar si dicho usuario tiene acceso al servicio o no.
- Permitir al propietario de una agenda, asignarle permisos de lectura, escritura o ambos, a otros usuarios; de manera que estos puedan leer o

quizá hasta añadir actividades en la agenda del usuario que ha otorgado los permisos respectivos.

- Permitirle al usuario ver aquellos mensajes que otros usuarios han grabado en la agenda que dicho usuario esté usando, en una hora determinada. Hay que anotar que el usuario puede ver estos mensajes ya sea sobre su propia agenda o sobre alguna otra en la cual este tenga los permisos pertinentes.

1.3. REQUERIMIENTOS DE RENDIMIENTO Y CONFIABILIDAD.

Para lograr que nuestra aplicación tenga un rendimiento aceptable y sea confiable en el mayor grado posible, necesitamos cumplir los siguientes requerimientos :

- Teniendo una linea de comunicación lo suficientemente rápida, se asegura que los tiempos de respuesta entre los requerimientos que se realicen entre el cliente y el servidor, sean los más cortos posibles. Sin embargo, el tiempo de procesamiento de los datos tanto en la máquina cliente como en la servidora, depéndera en este caso de la velocidad de procesamiento de cada una de dichas máquinas.
- Si en algún momento la conexión falla o se interrumpe, se garantiza que tanto el programa cliente como el servidor no se cancelen o queden inhibidos. Para lograr esto se hace uso de mecanismos tales como tiempos máximos de espera entre envio y recepción, para que de esta manera, si no

se recibe la respuesta en un tiempo máximo, se genere el mensaje de error correspondiente sin afectar a las aplicaciones.

- Si existiese algún error en la aplicación cliente o en la servidora, esto no afectará la conexión, en todo caso, si el error se produjera como respuesta a una transacción errada, el error será notificado a quien hizo el requerimiento.

Estos requerimientos se definen de esta manera dado que se ha pensado que la aplicación en general usará el esquema Cliente - Servidor.

CAPITULO II

2. DISEÑO DEL PROTOCOLO.

2.1 ARQUITECTURA CLIENTE SERVIDOR DE LA APLICACION.

Dado que una agenda puede ser usada por una sola persona o poder ser vista o escrita por otros usuarios, esto nos da la idea de que la agenda en realidad es una agenda de grupo, en la cual un administrador de la agenda o, jefe de grupo puede escribir y leer en las agendas de todos los usuarios que le permitan hacerlo. Como la agenda entonces, va a servir en realidad a un grupo de usuarios, se tiene que los datos de las agendas personales de cada uno de estos deben estar almacenados en archivos que tienen que estar en un mismo computador.

De lo anterior deducimos que los datos entonces, son centralizados, si queremos que el ingreso de actividades en la agenda sea usando una interfase amigable, y, además no se requiera que se ingresen estos datos desde el mismo computador en el cual se encuentran los archivos que contienen las agendas de los usuarios; se necesita entonces de un esquema que permita comunicar la aplicación de ingreso y formateo de datos por parte del usuario con aquella que maneja los archivos de las agendas de los usuarios.

Un esquema de programación que permita una interacción de esta clase, es el esquema Cliente - Servidor. De esta forma podemos tener una aplicación cliente que use una interfase amigable, así como un mecanismo de edición y de ingreso de datos que es fácil de manejar y eficiente, para el usuario. La aplicación servidora por otro lado puede estar ejecutándose libremente en la máquina que contendría los archivos de

las agendas, pudiendo ella de esta manera atender a las transacciones del cliente de manera transparente al usuario.

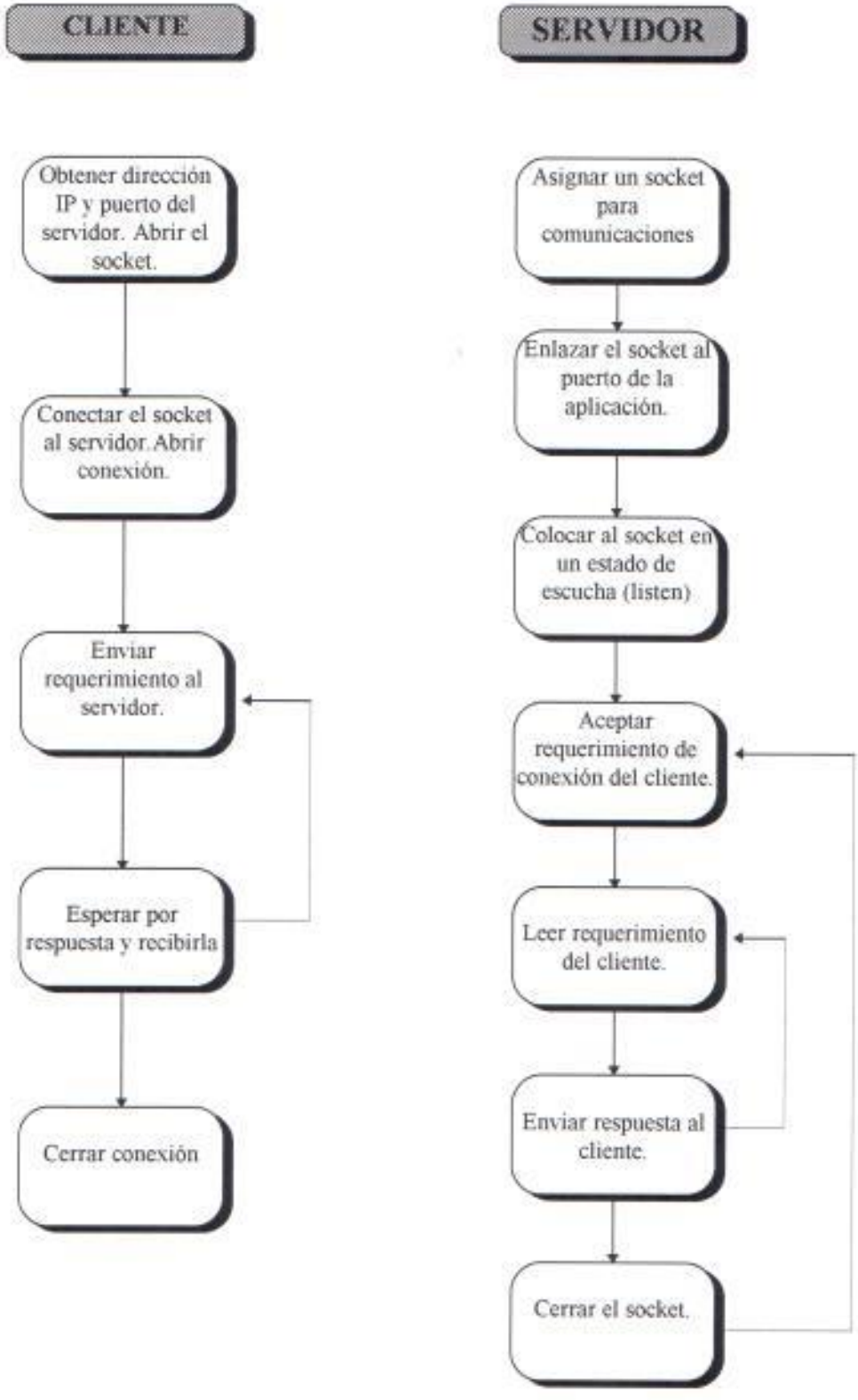
El esquema Cliente - Servidor, contiene obviamente dos tipos de aplicaciones. Un servidor el cual siempre está ejecutándose indefinidamente, además se tiene la aplicación cliente, la cual envía requerimientos al servidor y después espera por la respuesta que le envíe este. Como se usa TCP/IP como el protocolo de comunicación de datos, necesitamos de algún mecanismo o de alguna interfase que nos permita enlazar tanto a la aplicación cliente como a la aplicación servidora con el protocolo TCP/IP; ese mecanismo al cual se hace referencia es la interfase de sockets.

Los sockets son en realidad un recurso del sistema operativo, que nos permite abrir y mantener un canal de comunicación TCP/IP entre cliente y servidor, para que a través de este puedan interactuar ambas aplicaciones. Como se necesita que las aplicaciones cliente y servidor se comuniquen entre sí, se usa para el efecto, lo que definimos como buffer; se tiene un buffer que el cliente transmite al servidor con el requerimiento que este quiera hacer, y se tiene además el buffer que el servidor transmite como respuesta a un requerimiento del cliente. El buffer en conclusión no es más que una cadena de caracteres que se transmite entre cliente y servidor, y la cual se envía dentro del paquete TCP.

La aplicación servidora tiene asignado un puerto, es a través de este puerto que el programa servidor puede recibir los requerimientos de los clientes, y además, cuando un cliente necesita enviar un requerimiento al servidor, debe conocer el puerto

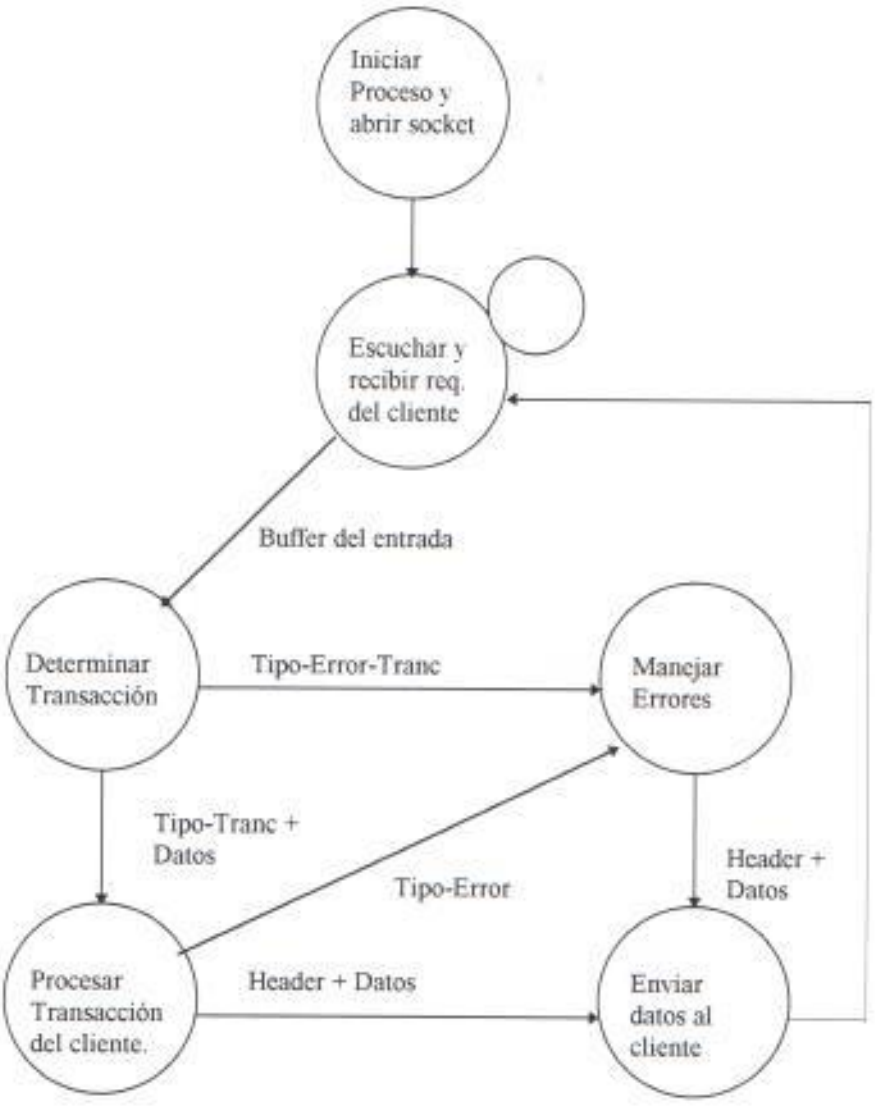
en el cual está el servidor. De manera similar el cliente también tiene un puerto y a ese puerto al cual se direcciona la respuesta del servidor al cliente.

A continuación se muestra un diagrama de como interactúan la aplicación cliente con la aplicación servidora.

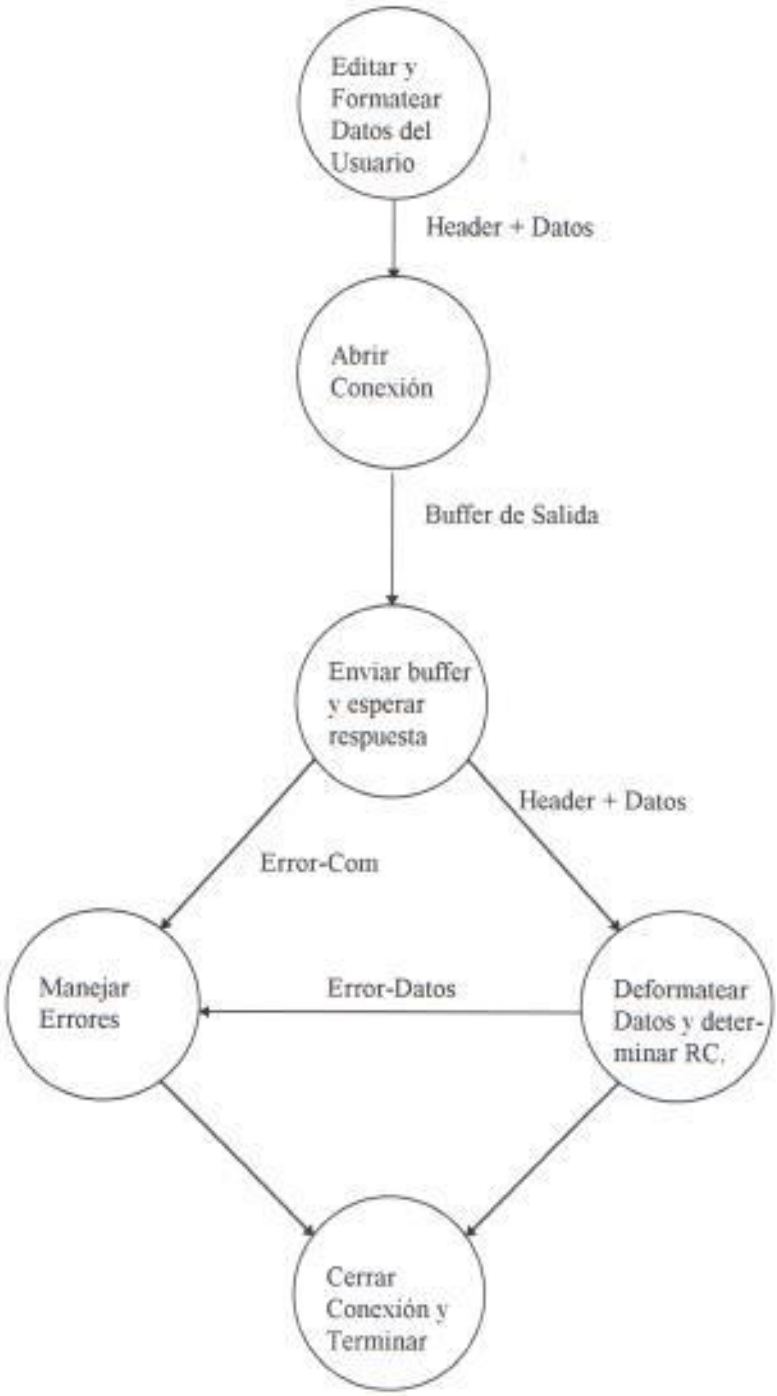

CLIENTE
SERVIDOR
Obtener dirección IP y puerto del servidor. Abrir el socket.
Conectar el socket al servidor. Abrir conexión.
Enviar requerimiento al servidor.
Esperar por respuesta y recibirla
Cerrar conexión
Asignar un socket para comunicaciones
Enlazar el socket al puerto de la aplicación.
Colocar al socket en un estado de escucha (listen)
Aceptar requerimiento de conexión del cliente.
Leer requerimiento del cliente.
Enviar respuesta al cliente.
Cerrar el socket.

2.2 MAQUINA DE ESTADOS DEL CLIENTE Y DEL SERVIDOR.

Máquina de estados del Servidor.



Máquina de estados del Cliente.



Para la máquina de estados del servidor se definen los siguientes estados :

- *Iniciar proceso y abrir socket* : En este estado, se inicializa el proceso servidor, se asigna el socket que la aplicación servidora usará para comunicarse con el cliente. Se enlaza el socket con el puerto asignado en el cual el proceso servidor recibirá los requerimientos del cliente.
- *Escuchar y recibir requerimientos del cliente* : En este estado, el proceso servidor está a la espera de que un requerimiento arribe al puerto en el cual el éste está ejecutándose. Una vez que llega un requerimiento de conexión desde el cliente, se crea un proceso mediante una llamada de sistema, el cual atenderá los requerimientos del cliente recién conectado.
- *Determinar transacción* : Del buffer recibido desde el cliente, se extrae la cabecera de éste, en el cual consta un campo que contiene el código de la transacción. Si la transacción existe, esta se ejecutará; si no existiese, se manejará la condición como un error.
- *Procesar transacción* : En este estado se procesa la transacción que ha solicitado el cliente, si ocurriese algún error en la operación, se maneja la condición de error, sino se preparará la respuesta a enviarse al usuario.
- *Manejar errores* : En este estado se formatea el buffer de respuesta al cliente cuando algún error ha ocurrido, los errores que maneja este estado tienen que ver con aquellos que podrian ocurren cuando alguna transacción no es ejecutada satisfactoriamente. En la cabecera del buffer que se envia al

cliente, se le coloca el código de retorno correspondiente al error que se ha generado.

- *Enviar datos al cliente* : En este estado se crea el buffer que será enviado al cliente como respuesta a un requerimiento hecho por este.

Además para la máquina de estados del servidor, se define lo siguiente :

- *Buffer de entrada* : Es el buffer que el servidor recibe del cliente, en el cual se contienen tanto la cabecera con la transacción a realizarce y los datos respectivos.
- *Tipo-Error-Tranc* : Es un código de error que genera el servidor, cuando una transacción enviada por el cliente no existe.
- *Tipo-Tranc+Datos* : Del buffer que llega del cliente, se extrae la transacción solicitada por el cliente (Tipo-Tranc), y los datos que envia el cliente (Datos).
- *Tipo-Error* : Dependiendo del error que ocurra cuando se ejecute alguna transacción en el servidor, éste genera el código de error correspondiente.
- *Header + Datos* : Se construye la cabecera (Header), el cual contiene los campos que forman el protocolo que se ha defindo para la aplicación, con los códigos de respuesta correspondientes a la transacción solicitada por el cliente; se le añaden los datos que el servidor retorne al cliente (Datos), si la transacción asi lo requiere.

Para la máquina de estados del cliente, se tienen los siguientes estados definidos :

- *Editar y formatear datos del usuario* : En este estado, la aplicación cliente le permite al usuario ingresar los datos que este desea procesar en el servidor. Una vez que se tengan todos los datos requeridos, se crea la cebecera del buffer que se enviará al servidor, colocándole la transacción respectiva y de acuerdo al protocolo definido.
- *Abrir conexión* : En este estado, la aplicación cliente se encarga de establecer la comunicación con la aplicación servidora a través del socket asignado.
- *Enviar buffer y esperar respuesta* : La aplicación cliente, una vez establecida la conexión, envia el buffer (cabecera y datos) al proceso servidor, y se queda en espera hasta que el servidor le responda.
- *Deformatear datos y determinar RC* : La aplicación cliente, una vez que ha recibido el buffer de respuesta desde el servidor, separa la cabecera de los datos. De la cabecera, se extrae el código de retorno (RC), y dependiendo de su valor, se sabrá si se debe manejar alguna condición de error o no. Con la parte correspondiente a los datos, la aplicación servidora separa los campos que se puedan contener en estos y son mostrados al usuario.
- *Manejar errores* : La aplicación cliente manejará dos tipos de errores que puedan ocurrie. El primer tipo tiene que ver con el código de retorno que recibe el cliente desde el servidor como respuesta a un requerimiento hecho por el propio cliente. El segundo tipo tiene que ver con aquel que puede generarse cuando ocurre un error en la comunicación con el servidor.

- *Cerrar conexión y terminar* : Una vez que se ha recibido el buffer desde el servidor, y se ha procesado este, se termina la conexión con el servidor y de esa manera también se termina la transacción. De esta manera se tiene que por cada transacción se debe siempre abrir y cerrar la conexión con el servidor.

Además para la máquina de estados del cliente se define lo siguiente :

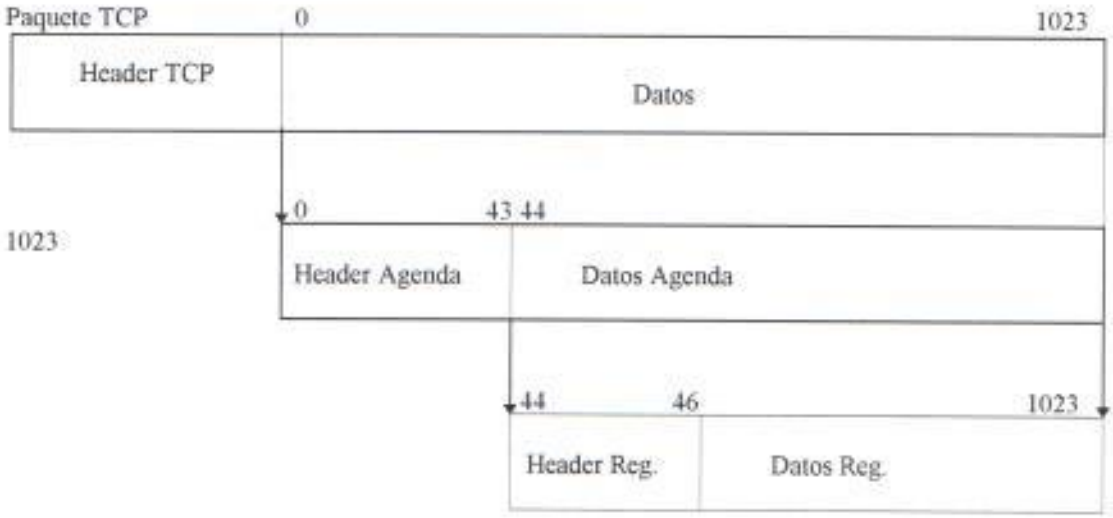
- *Header + Datos* : De los datos que el usuario ingrese a la aplicación cliente, ésta construye la cabecera (Header), con la transacción correspondiente y le añade los datos (Datos) que el usuario haya ingresado.
- *Buffer de Salida* : Con la cebecera y los datos (Header + Datos), la aplicación cliente construte el buffer que se enviará a la aplicación servidora.
- *Error - Com* : Tipo de error que se genera cuando se ha producido algún problema en la comunicación.
- *Error - Datos* : Tipo de error retornado por el servidor al cliente cuando alguna transacción no se ha realizado correctamente , o cuando los datos que se retornaron al cliente no han sido los esperados.

2.3 SINTAXIS Y SEMANTICA DEL PROTOCOLO EN EL QUE SE BASA EL CLIENTE Y EL SERVIDOR.

Nuestra aplicación (tanto cliente como servidor), debe ejecutar básicamente dos actividades diferentes, una de ellas es la dedicada al ingreso de usuarios

autorizados al servicio asi como su autenticación, y, la otra es la que se encarga del manejo de los archivos de la agenda de cada usuario, es decir, se encarga de hacer los ingresos, modificaciones, eliminaciones y adiciones de actividades en la agenda de cada usuario.

El protocolo de manejo de transacciones de la agenda, se lo podria representar de la siguiente manera :



Como se puede observar en el diagrama, dentro del paquete TCP, la parte de los datos, contiene el buffer de nuestra aplicación. El buffer contiene el *Header Agenda*, que es la cabecera del buffer que usará nuestra aplicación, y en la que se define el protocolo a usarce. En *Header Reg*, se contiene una pequeña cabecera que se ha definido por registro, esta sirve a los registros correspondientes a las actividades de la agenda y a los registros de usuarios.

La cabecera Header Reg, se usa para realizar operaciones de ingreso, eliminación o modificación por registro enviado en la parte de datos. Con esta cabecera la aplicación podrá ser capaz de realizar operaciones individuales por cada registro que se envíe.

El protocolo (contenido en la cabecera general), se ha definido de la siguiente manera :

CODTRAN	Código de transacción (3 bytes)
USUARIO	Login name del usuario que efectúa la transacción (8 bytes)
AGENDA	Archivo de agenda utilizado (8 bytes)
PASSWORD O NAMEFILE	Password : Password del usuario Namefile : Archivo para consulta de actividades (8 bytes)
CLAVE_ALTERNA	Para búsqueda sucesivas de registros (8 bytes)
LONG_RESP	Indica la cantidad en bytes del (2 bytes) buffer enviado desde el servidor
REG_REQ	Indica cuántos registros solicita el cliente (2 bytes)

REG_RET	Indica cuántos registros ha retornado el servidor (2 bytes)
MAS_REG	Le indica al cliente si el servidor tiene mas registros por enviarle (2 bytes)
RC	Código de retorno del servidor al cliente (2 bytes)

Para la cabecera que tiene por cada registro a enviarse dentro de la parte de datos del buffer de la aplicación, se tiene que enviar junto con los datos correspondientes a cada registro el código de transacción correspondiente a dicho registro, para que de esa manera la aplicación servidora pueda determinar el tipo de operación que realizará con cada uno de esos registros.

CAPITULO III

3. DISEÑO DEL SERVIDOR.

3.1. FUNCIONALIDADES DEL SERVIDOR.

El servidor de encargará básicamente de efectuar las siguientes operaciones :

- Efectuar la autenticación para el acceso al servicio por parte de algún usuario, es decir, cuando la aplicación cliente le envíe en un buffer el nombre del usuario y el password respectivo acompañado de la transacción correspondiente. La aplicación servidora se encargará de verificar si dicho usuario está contenido en el archivo de usuarios autorizados al servicio, si el usuario no existiese se le enviará el mensaje de error correspondiente al cliente.
- Aceptar transacciones de Ingreso, Modificación, Consulta y Eliminación de datos (actividades) de la agenda, para el usuario propietario de la agenda. Ser capaz, además, de poder efectuar la adición de datos a alguna agenda por parte de usuarios que tengan permiso para ello, los datos añadidos por un usuario no pueden ser modificados por otro que también tenga permisos sobre esa misma agenda.
- Modificar los archivos que contengan los permisos para aquellos usuarios que puedan leer y escribir en la agenda de otro usuario, esto se hace como respuesta a un requerimiento del cliente, donde el propietario de la agenda es el que otorga los permisos a aquellos usuarios a quienes desee dárselos.

- Poder ingresar y eliminar usuarios autorizados al servicio de agenda, se hará interactuando con el servidor mismo a través del programa cliente, o a través de un programa de administración que se encuentra en el mismo computador en el que corre el servidor. Sólo el administrador del servidor de la aplicación puede crear o eliminar usuarios autorizados al servicio ofrecido.

3.2. TIPO DE SERVIDOR Y SU JUSTIFICACION.

Cuando un usuario utiliza la aplicación cliente para efectuar ya sea un ingreso de actividades propiamente dicho o una asignación de permisos para otros usuarios sobre su agenda, por ejemplo; el cliente deberá enviar un buffer de datos que no será siempre de la misma longitud en cada iteracción que este haga. Si se requieren consultas repetidas o ingresos sucesivos asi como también modificaciones, hay que recordad que todas estas operaciones las realiza en realidad el programa servidor sobre los archivos destinados a almacenar los datos de las agendas para cada usuario.

Las operaciones que realiza el servidor y que describimos en resúmen en el párrafo anterior, no son operaciones que se realicen en un tiempo relativamente corto, además es muy probable que la aplicación cliente, esté corriendo en un computador que no posea la caracteristica de ser multitarea, de esta forma, si cada requerimiento se realiza después de que termina el anterior, o sea, iterativamente; el usuario en la máquina que corre la aplicación cliente deberá esperar a que su requerimiento sea aceptado por el servidor, procesado y después recibir la respuesta.

Si consideramos todo lo dicho anteriormente y si pensamos por un momento que el computador que corre la aplicación servidora puede estar sujeto a una gran carga de trabajo, entonces, la aplicación cliente deberá esperar un tiempo quizá relativamente grande para que su requerimiento sea atendido, sin contar con el tiempo que podria demorar el procesamiento del requerimiento en el servidor, esta situación haria que los usuarios eviten usar la aplicación cliente, ya que les demandaria mucho tiempo de espera, el cual podrian aprovechar efectuando alguna otra operación en el computador que corre la aplicación cliente, y que no lo pueden hacer, porque si el cliente corre en un computador personal monotarea, se deberia terminar un proceso para efectuar el siguiente.

Si la aplicación cliente está corriendo en un computador con un sistema operativo multitarea, que sea compatible con una arquitectura PC, tal como OS/2, por ejemplo; quizá podria sacrificarse un poco y dejar su aplicación cliente esperando algún tiempo y hacer alguna otra cosa.

Considerando todo estas situaciones, lo más lógico si tenemos un computador con un sistema operativo multitarea, en el que corre la aplicación servidora, es que tengamos un servidor que pueda atender varios requerimientos simultáneamente. Esto se logra teniendo un servidor **concurrente multiproceso**. Este tipo de aplicación permite crear un proceso esclavo por cada conexión que se tenga con algún cliente dado, de forma que cada requerimiento se maneja en forma separada como un proceso aparte.

Como el cliente usa códigos de transacción relacionados con cada operación que se realiza, decimos que nuestro servidor es **multitransaccional**. Cada vez que un usuario ingresa (efectúa un proceso de logon) a la aplicación, se registra en un archivo creado para el caso al usuario que ha ingresado, esto se hace debido a que como la conexión sólo existe cuando se efectúa una transacción, se necesita saber si el usuario ha sido autenticado o no cada vez que éste solicite que se ejecute alguna transacción. Por esta razón, debido a que el servidor mantiene este registro podemos decir también que nuestro programa servidor es **Stateless**, debido a que no se mantiene un control del estado de la conexión.

3.3. DISEÑO DE LOS DATOS MANEJADOS EN EL SERVIDOR.

De acuerdo con el protocolo especificado, a través del cual las aplicaciones cliente y servidor se comunican, tenemos las siguientes estructuras definidas:

- Para la representación del protocolo en el cual se basará la clomunicación entre el cliente y el servidor.

NOMBRE	TIPO	LONGITUD (BYTES)
codtran	caracter	9
usuario	caracter	9
agenda	caracter	9
password / file	caracter	9
len_req	entero	4

len_resp	entero	4
qt_reg_req	entero	4
qt_reg_ret	entero	4
mas_reg	caracter	1
rc	entero	4

- Para el manejo de los registros correspondientes a operaciones de ingreso, modificación, eliminación, etc, de usuarios, se define la siguiente estructura :

NOMBRE	TIPO	LONGITUD (BYTES)
codtran	caracter	4
usuario	caracter	9
password	caracter	9
desc	caracter	31

- Para el manejo de los registros correspondientes a operaciones de ingreso, modificación, consulta, o eliminación de actividades en la agenda, se define la siguiente estructura :

NOMBRE	TIPO	LONGITUD (BYTES)
codtran	caracter	4
fecha	caracter	8
hora	caracter	6
alarma	caracter	1
desc	caracter	200

- Se ha definido también en el servidor el archivo **Logon**, el cual lleva el registro de los usuarios autorizados al servicio de la agenda que han tenido acceso al sistema, la estructura del archivo se muestra a continuación :

NOMBRE	TIPO	LONGITUD (BYTES)
usuario	caracter	8
horain	caracter	8
horaout	caracter	8

- El archivo **Usuarios** contiene los datos referentes a los usuarios que tienen acceso al servicio de la agenda, este archivo contiene el login name del usuario, el password y la descripción de cada usuario. La estructura del archivo se muestra acontinuación :

NOMBRE	TIPO	LONGITUD (BYTES)
usuario	caracter	8
password	caracter	8
descripción	caracter	30

- El archivo de Permisos contiene información respecto de los permisos que un usuario otorga a otros usuarios sobre su agenda. Los campos de este archivo son el login name del usuario al cual se le han otorgado los permisos, el nombre de la agenda sobre la cual se ha otorgado el permiso, los permisos (lectura ,escritura, o administración). Cuando nos referimos al permiso de administración, hacemos en realidad referencia al usuario que es el administrador de la agenda, por lo tanto sólo existe un usuario con este atributo. La estructura del archivo es la siguiente :

NOMBRE	TIPO	LONGITUD
usuario	caracter	8
agenda	caracter	8
lectura	caracter	1
append	caracter	1
adm	caracter	1

- El servidor utiliza también un archivo de configuración (sagenda.cfg), en el cual se fijan ciertos parámetros que el servidor usa para poder funcionar sin problemas. Los parámetros pueden ser alterados de manera que el proceso servidor pueda actuar de manera diferente, el archivo lo podemos ver en el apéndice 2.

En el apéndice 5 podemos ver la definición de las estructuras citadas en este apartado, hechas en lenguaje C, que ha sido el lenguaje en el que se programó la aplicación servidora.

3.4. DISEÑO DE LA APLICACION SERVIDORA.

La aplicación servidora está basada en procedimientos, es decir se define un programa principal y diferentes funciones y procedimientos que se definen para efectuar todas las operaciones que realice el programa servidor, incluyendo las transacciones que se realizan entre cliente y servidor. Cada uno de estos procedimientos o funciones se definen en algoritmos que se detallan a continuación.

3.4.1. Algoritmo PRINCIPAL.

Este algoritmo describe el funcionamiento del proceso servidor en si, sin entrar en detalles de las operaciones que se efectúan para cada transacción dada.

1.- Crea un Socket y lo asocia a un puerto conocido.

2.- Coloca el Socket en modo pasivo.

3.- Repetidamente llama a accept, para recibir el próximo requerimiento de un cliente.

4.- Crea un proceso esclavo para manejar el buffer que llega desde el cliente.

1.1 Recibe un requerimiento de conexión, se usa esta conxión para iteractuar con el cliente.

1.2 Recibe el buffer y determina el tipo de transacción, si la transacción no es válida, se le envía un código de error al cliente.

1.3 Ejecuta la transacción y responde al cliente.

1 Cierra la conexión una vez que se han satisfecho los requerimientos.

3.4.2. Algoritmo LOGON.

A través de este algoritmo podemos autenticar el login name y el password del usuario que desea acceder al servicio de la agenda.

1.- Recibe el buffer del cliente y extrae la estructura que contiene el login name y el password del usuario.

2.- De la estructura extraida en el punto 1, se obtienen el login name y el password del usuario.

3.- El login name y el password del usuario son comparados contra el archivo que contiene a todos los usuarios autorizados al servicio.

4.- Si el usuario existe, se crea una entrada en un archivo que contiene el usuario que ha accesado al servicio, este archivo se crea para registar si el usuario que efectúa alguna transacción, ya ha accedido al servicio primero.

5.- Si el usuario no existe, se envia el código de error respectivo al cliente.

3.4.3 Algoritmo LOGOFF.

Este algoritmo explica el proceso de desconexión o eliminación del registro de acceso al servicio de la agenda.

1.- Dada la transacción de LOGOFF, se extrae del buffer del cliente el login name del usuario.

2.- Con el login name del usuario, se verifica que éste se encuentre registrado en el archivo de usuarios que han dado LOGON, si se encuentra allí, se elimina la entrada correspondiente a ese usuario.

3.- Si no se encuentra en dicho archivo, se le envía el código de error respectivo al cliente.

3.4.4. Algoritmo ING_USR.

Este algoritmo explica el proceso de ingreso de un nuevo usuario autorizado al servicio de la agenda.

1.- Del buffer que llega del cliente, se extrae la estructura que contienen los datos del nuevo usuario a ingresar.

2.- Se crea un directorio en el cual se colocarán las actividades que el usuario coloque en su agenda.

3.- Se le creará una entrada en el archivo que contiene los usuarios autorizados.

4.- Se le creará una entrada en el archivo de permisos sobre la agenda de propiedad del nuevo usuario.

5.- Se envía un código de retorno al cliente.

3.4.5. Algoritmo CON_USR.

Este algoritmo explica el proceso de consulta de usuarios que se realiza en el servidor, hay que anotar que esta consulta es masiva, es decir; cada vez que se efectúa una transacción de consulta al servidor, éste le envía todos los usuarios con acceso al servicio de la agenda.

1.- Se abre el archivo que contiene los usuarios con acceso al servcio de la agenda.

2.- Se lee registro por registro dicho archivo, y se llena el buffer en el cual se le enviarán los datos al cliente.

3.- Si se llena el buffer de envío, se lo envía con una marca indicando que existen más registros por enviarse.

4.- Si existen más registros por enviarse, el cliente envía otra transacción y el servidor envía los siguientes registros, se regresa al paso 3 si existen más registros, sino se termina la consulta.

3.4.6. Algoritmo MOD_USR.

Este algoritmo ilustra cómo el servidor realiza una modificación de datos de un usuario, esta operación se la realiza para un usuario a la vez.

1.- Del buffer de llegada del cliente, se extrae el registro que contiene el login name del usuario a modificarse, junto con sus datos.

2.- Se escriben los datos del usuario en el archivo que contiene a los usuarios autorizados al servicio de la agenda.

3.- Se envía el código de retorno al cliente.

3.4.7. Algoritmo ELI_USR.

A través de este algoritmo, se explica cómo el servidor elimina a un usuario del archivo de usuarios.

1.- Del buffer de llegada del cliente, se extrae el registro que contiene el login name del usuario a eliminarse.

2.- Se elimina la entrada de ese usuario del archivo de usuarios autorizados.

3.- Se eliminan los archivos donde aquel usuario mantiene las actividades de su agenda.

4.- Se borran todos los permisos que este usuario tenga sobre otras agendas.

5.- Se envía código de retorno al cliente.

3.4.8. Algoritmo ING_ACT.

Este algoritmo explica cómo el servidor ingresa o añade una nueva actividad en la agenda del usuario que efectúa esta transacción.

1.- Del buffer de llegada del cliente, se extraen todos los registros correspondientes a actividades a ingresarse en la agenda.

2.- Cada registro de actividad enviado por el cliente, es ingresado en el archivo que corresponde a la fecha en la que se ingresa la actividad; cada archivo que corresponde a una fecha diferente se encuentra en el directorio de la agenda del usuario.

3.- Si existen más actividades por ingresarse, el cliente envía otro buffer lleno de registros de actividades, repitiéndose los puntos 1 y 2.

4.- Una vez finalizada la transacción, se le envía un código de retorno al cliente.

3.4.9. Algoritmo CON_ACT.

Este algoritmo explica cómo el servidor realiza una consulta de actividades de una agenda dada.

1.- Del buffer de llegada del cliente, se extrae el registro que contiene la fecha de la agenda cuyas actividades se quieren consultar.

2.- Con la fecha que se obtiene, se abre el archivo correspondiente a la fecha que se quiere consultar.

3.- Se extraen las actividades que se encuentren en ese archivo para la fecha que se está consultando, hasta que se llene el buffer que se le va a enviar al cliente.

4.- Si existen más registros por envierse, el servidor le envía una seña al cliente indicando que existen más registros.

5.- Cuando el cliente se percata de que existen más registros por recibir, efectúa otra transacción, y el servidor le envía los registros restantes repitiendo el paso 3.

6.- Si no existen más registros, se le notifica esto al cliente, de manera que no realice ninguna otra transacción de consulta de actividades.

3.4.10. Algoritmo MOD_ACT.

Este algoritmo explica cómo el programa servidor modifica alguna actividad previamente ingresada en la agenda.

1.- Del buffer de llegada del cliente, se extraen los registros de actividades a modificar.

2.- Se abre el archivo que corresponde a la fecha que contiene las actividades que se van a modificar.

3.- Por cada actividad que se encuentre en el buffer de llegada del cliente, ésta es modificada en el archivo correspondiente.

4.- Si el cliente tiene más actividades que enviar, formatea otro buffer con registros de actividades y lo envía al servidor, repitiéndose desde el paso 1.

5.- Una vez que se modifiquen las actividades que se han enviado al servidor, éste le envía un código de retorno al cliente..

3.4.11. Algoritmo ELI_ACT.

Este algoritmo explica cómo el servidor elimina alguna actividad de la agenda.

1.- Del buffer de llegada del cliente se extraen los registros que contienen las actividades a eliminarse.

2.- Por cada registro en el buffer que contiene alguna actividad a eliminarse, éste es eliminado del archivo que contiene dichas actividades.

3.- Una vez que el servidor ha efectuado la transacción , le envia un código de retorno al cliente.

3.4.12. Algoritmo PERMISOS.

Este algoritmo explica cómo un usuario puede otorgar permisos a otros usuarios para que puedan leer y hasta añadir actividades sobre la agenda de la cual dicho usuario es propietario.

1.- Del buffer de llegada del cliente se extrae el registro que contiene los datos necesarios.

2.- En el archivo donde se almacenan los registros otorgados a los usuarios, se ingresa el login name del usuario a quien se le otroga permisos, la agenda sobre la cual se le dan los permisos, y, los permisos que se están otorgando.

3.- Una vez que el servidor termina la transacción, se envia un código de retorno al servidor.

3.5 COMPROMISOS DEL DISEÑO.

- Aunque se tiene un servidor concurrente, existe de todas maneras una cola de espera de requerimientos que arriban al puerto a través del cual el servidor con el cliente, de esta manera si llega un número de requerimientos mayor al tamaño de la cola, éstos no son tomados en cuenta por el servidor. Los procesos se encolan debido a que el proceso servidor no

puede atender a un cliente si se encuentra efectuando un fork para crear un proceso hijo.

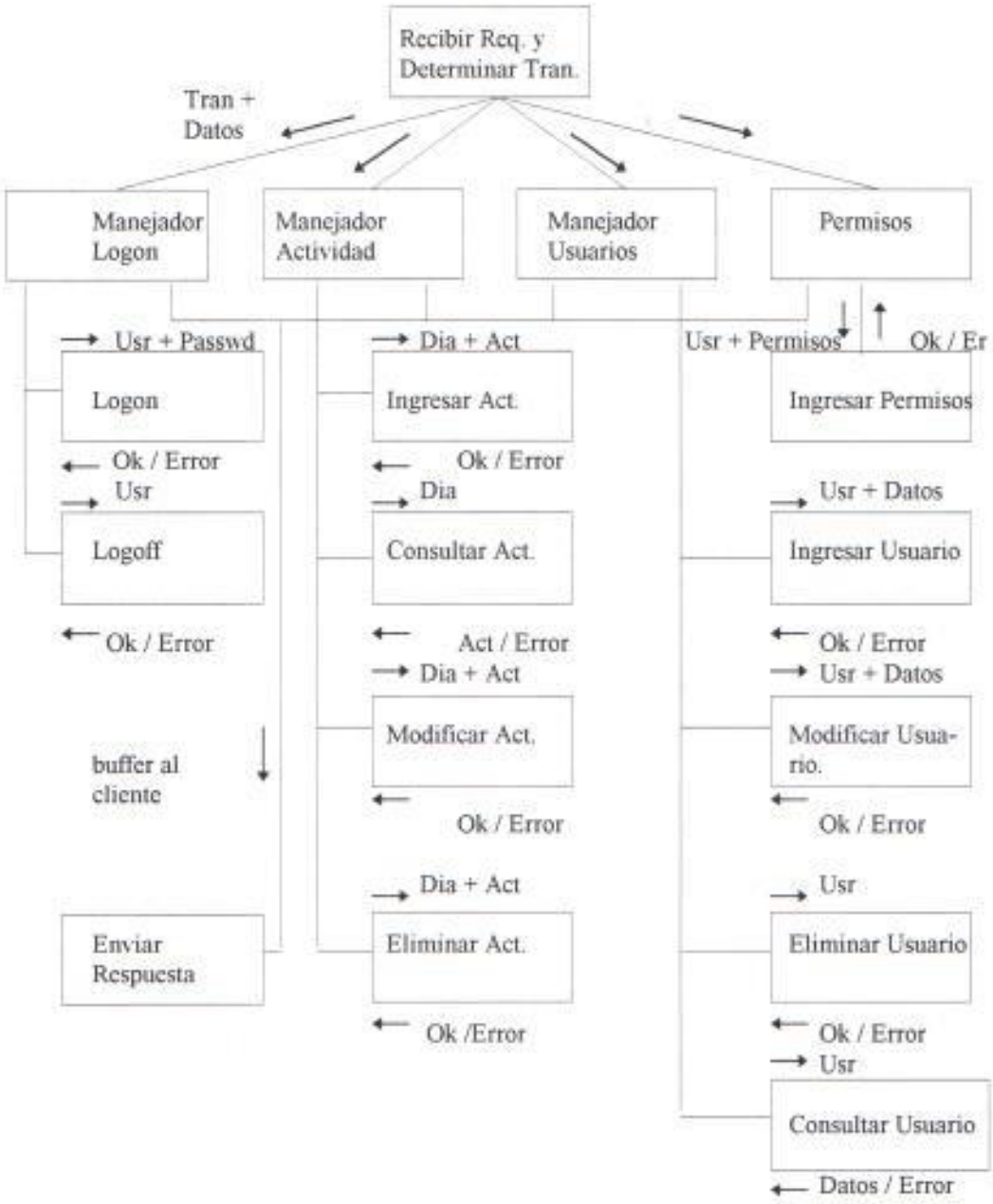
- Aunque un servidor concurrente puede garantizar la ejecución de requerimientos de manera simultánea y eficiente, esto no puede asegurarnos que se introduzcan demoras en el procesamiento debido a problemas de transmisión que pueden ser tales como : congestión en la línea, velocidad baja de transmisión, ruido en la línea, etc.
- Debido a que la aplicación servidora se comunica con la aplicación cliente usando un protcolo TCP/IP, no nos preocupamos de la integridad y retransmisión de los datos, ya que el control de errores y de flujo está cubierto por el protocolo mismo.
- La aplicación servidora crea un proceso esclavo por cada requerimiento de un cliente dado, teniendo en cuenta que cada cliente puede efectuar un requerimiento a la vez. No se garantiza de ninguna manera, sin embargo, que si un requerimiento de algún cliente es atendido en un lapso pequeño de tiempo, el siguiente será tratado de la misma manera.

3.6 ADMINISTRACION DEL SERVIDOR

Esistirá un usuario que va a tener los atribitos del administrador, él será el responsable de configurar mediante el archivo correspondiente los parámetros de inicialización del servidor. Otra responsabilidad del administrador, será el mantenimiento del archivo que contiene los usuarios con acceso al servicio, sólo el administrador podrá crear o eliminar usuarios.

El mantenimiento del archivo de usuarios, lo puede hacer el administrador ya sea usando el programa cliente, el cual presta esta facilidad, o también, a través de un programa de administración hecho para tal efecto, el cual se encuentra en la misma máquina donde corre la aplicación servidora. El programa de administración, le permite también al administrador eliminar registros físicamente de los archivos de atributos y de usuarios, dado que cuando se elimina algún registro de cualquiera de estos archivos, la eliminación es lógica, a través del programa administrador podemos eliminar físicamente esos registros.

3.7. DIAGRAMA DE BLOQUES DE LOS PROCEDIMIENTOS DEFINIDOS EN EL SERVIDOR.



CAPITULO IV

Agenda ESPOL v1.0

4. DISEÑO DEL CLIENTE.

4.1 FUNCIONALIDADES DEL CLIENTE.

La aplicación Cliente tendrá las siguientes funcionalidades :

- Establecer la conexión con la aplicación servidora, y una vez que los requerimientos sean atendidos, cerrar la conexión.
- Permitir al usuario ingresar su nombre y password, crear el buffer para transmisión y enviar éste al servidor, el servidor se encarga de hacer la autenticación.
- Permitir al usuario a través de una interfase amigable y de fácil manejo, editar las actividades a ingresar o modificar en su agenda. Crea el buffer con la respectiva transacción para que sea procesada por la aplicación servidora.
- Permitir al usuario asignarle permisos a otros usuarios para que puedan ver y quizá también añadir actividades en la agenda de su propiedad.
- El cliente, al recibir los datos desde la aplicación servidora, debe identificar el tipo de mensaje que ha recibido y hacerle saber al usuario si son respuestas a requerimientos o códigos de error que se han generado.
- El cliente estará encargado de manejar los códigos de error ya sea que los haya generado él mismo o que provengan desde el servidor. El cliente le mostrará al usuario el mensaje de error correspondiente y el código de error que se ha generado.

- El cliente proveerá un sistema de ayuda, de manera que el usuario pueda orientarse mejor en el uso de la aplicación.

4.2 DISEÑO DE LOS DATOS MANEJADOS EN EL CLIENTE.

Dado que el cliente ha sido implementado en Visual Basic 3.0 para Windows, tenemos las definiciones de los principales datos a usarce, como se explica a continuación.

- Para la comunicación entre el cliente y el servidor, se define la siguiente estructura que contiene la cabecera del buffer a enviarse al servidor :

NOMBRE	TIPO	LONGITUD (BYTES)
codtran	string	4
usuario	string	9
agenda	string	9
clave_alterna	string	9
tipo_busqueda	string	1
clave_bsq	string	6
len_req	string	4
qt_reg_req	string	4
len_resp	string	4
qt_reg_ret	string	4

mas_reg	string	1
rc	string	4

- Para los registros de manejo de usuarios para operaciones de ingreso, consulta, modificación, y, eliminación, se define la siguiente estructura :

NOMBRE	TIPO	LONGITUD (BYTES)
codtran	string	4
Usuario	string	9
password	string	9
desc	string	31

- Para el manejo de los registros correspondientes a los permisos que un usuario otorga a otros sobre su agenda, se define la siguiente estructura :

NOMBRE	TIPO	LONGITUD (BYTES)
codtran	string	4
Usuario	string	9
agenda	string	9
lectura	string	1
escritura	string	1
adm	string	1

- Para manejar los registros correspondientes a las actividades que un usuario puede ingresar, modificar, consultar o eliminar en su agenda o en las agendas a las que el usuario pueda acceder. Se define la siguiente estructura :

NOMBRE	TIPO	LONGITUD (BYTES)
codtran	string	4
hora	string	6
alarma	string	1
desc	string	200

La última estructura definida representa los regisrtos de las actividades de una agenda por hora, el dia de la agenda consultada, es introducida en la cabecera de datos cuando se ejecuta alguna transacción relativa a la agenda con el servidor. La definición de las estructuras y de las constantes usadas en el programa cliente, se incluyen en el apéndice 6.

4.3 DISEÑO DE LA APLICACION CLIENTE.

La aplicación cliente se ha diseñado basándose en la definición de objetos utilizados en Windows, tales como botones, cajas de edición de texto, ventanas en las cuales poder ingresar y mostrar datos ingresados por el usuario, menús tipo pop-down y de ventana, etc; para estos objetos se definen procedimientos relacionados, los cuales se ejecutan cuando ocurre algún evento en dicho objeto, como por ejemplo, cuando se presiona algún botón, se ejecuta el código relacionado con la operación relacionada a dicho botón. A través de la herramienta de programación que se ha utilizado para codificar el programa cliente, se pueden diseñar los objetos Windows, que se definen para la aplicación.

Los algoritmos que se detallan a continuación muestran la secuencia de pasos que siguen los procedimientos definidos y que están relacionados a los eventos que ocurren una vez que el usuario ha ingresado los datos requeridos.

4.3.1. Algoritmo SEND_LOGON.

Este algoritmo explica cómo se realiza el proceso de Logon desde el cliente hacia el servidor.

1.- Editar el login name y el password del usuario.

2.- Formar la cebecera del buffer de datos con la transacción correspondiente, y formatear los datos ingresados en el buffer a enviarse al servidor.

3.- Envia el buffer al servidor, y espera respuesta.

4.- Una vez recibido el buffer de respuesta del servidor, extraer el código de retorno recibido de éste.

5.- Si el código fué exitoso, habilitar las transacciones de manejo de la agenda, y, guardar el login name del usuario en una variable global; sino enviar el mensaje de error correspondiente.

4.3.2. Algoritmo SEND_LOGOFF.

Este algoritmo explica cómo el cliente realiza el proceso de logoff.

1.- Si el usuario ha dado logon, se extrae el login name del usuario de la variable en donde se lo almacenó, cuando éste realizó un logon.

2.- Formar la cabecera de buffer de datos con la transacción correspondiente y formatear los datos a en el buffer a enviarse al servidor.

3.- Enviar al buffer al servidor y esperar la respuesta.

4.- Una vez recibido el buffer de respuesta del servidor, extraer el código de retorno recibido de éste.

5.- Si el código fué exitoso, se dehabilitan las las transacciones de manejo de la agenda, si el código no fué exitoso, enviar el mensaje de error respectivo al usuario.

4.3.3. Algoritmo SEND_ING_USR.

Este algoritmo explica cómo el usuario, en este caso el administrador de la agenda, puede crear usuarios usando la aplicación cliente.

1.- Ingresar los campos correspondientes al login name, password y descripción del nuevo usuario a crear.

2.- Formar la cebecera del buffer de datos con la transacción respectiva y formatear los datos ingresados en el buffer a enviarse al servidor

3.- Enviar el buffer al servidor y esperar la respuesta.

4.- Una vez recibido el buffer de respuesta desde el servidor, se extrae el código de retorno enviado de éste al cliente.

5.- Dependiendo del código de retorno, se muestra el mensaje respectivo al usuario.

4.3.4. Algoritmo SEND_CON_USR.

Este algoritmo explica cómo un usuario puede realizar una consulta de usuarios aurorizados al servicio de agenda. La consulta que se realiza es masiva, es decir cada vez que se realiza esa transacción, se obtienen todos los usuarios autorizados desde el servidor.

1.- Formar la cebecera con la transacción respectiva de consulta y enviar el buffer al servidor.

2.- Enviar el buffer al servidor y esperar la respuesta.

3.- Una vez llegado el buffer desde el servidor, extraer el código de respuesta de éste.

4.- Si el código de retorno ha sido exitoso, se extraen los registros de los usuarios que se encuentren en el buffer que envió el servidor y se los coloca en un arreglo. Si el código de retorno no fué exitoso, se envia el mensaje respectivo al usurio.

5.- Si el servidor le envió al cliente una seña de que existen más registros por enviar, el cliente formatea un nuevo buffer de envio con el último registro recibido en el buffer y forma la cabecera con la transacción, y efectuamos los pasos definidos a partir del 2.

6.- Cuando el servidor envie una seña indicando que ya no existen más registros, se muestra el arreglo de regsitros de usuarios, consultados desde el servidor, al usuario que está efectuando la transacción.

4.3.5. Algoritmo SEND_MOD_USR.

Con este algoritmo se explica la forma en la cual el programa cliente hace la modificación de los datos de un usuario dado en el archivo de usuarios en el servidor. Esta transacción sólo puede ser realizada por el administrador de la agenda.

1.- Efectuar una transacción de consulta de todos los usuarios autorizados al servicio. Esto ya se explicó en el algoritmo anterior.

2.- Una vez que se tenga la lista que contiene a todos los usuarios, se escoge uno de ellos.

3.- Para el usuario que se ha escogido, se le pueden modificar los campos de password y de descripción.

4.- Una vez efectuados los cambios, se forma la cabecera del buffer con la transacción correspondiente,y se formatean los datos en el buffer para enviarlos al servidor.

5.- Enviar el buffer al servidor y esperar por la respuesta.

6.- Del buffer enviado desde el servidor, extraer el código de retorno. Dependiendo del valor de dicho código, se muestra el mensaje correspondiente.

4.3.6. Algoritmo SEND_ELI_USR.

Este algoritmo explica el proceso que sigue el programa cliente para eliminar un usuario del archivo de usuarios que se encuentra en el servidor. Esta transacción sólo puede ser realizada por el administrador de la agenda.

1.- Hacer una transacción de consulta de todos los usuarios autorizados.

2.- Una vez que se tenga la lista de todos los usuarios obtenidos desde el servidor, escoger uno para eliminación.

3.- Mostrar los datos del cliente y esperar por confirmación de eliminación.

4.- Una vez se confirme la eliminación, se forma la cabecera del buffer con la transacción correspondiente, se formatean los datos referentes al usuario a eliminarse en el buffer de envio al servidor.

5.- Enviar el buffer al servidor y esperar respuesta.

6.- Una vez que se tenga el buffer enviado por el servidor, obtener de éste el código de retorno, y de acuerdo al valor de dicho código mostrar el mensaje correspondiente al usuario.

4.3.7. Algoritmo ING_ACT.

Este algoritmo describe cómo el cliente realiza un ingreso de actividades en la agenda que el usuario está usando.

1.- Ingresar la fecha en la cual se desee añadirse alguna actividad.

2.- Por cada actividad a añadirse, ingresar la hora y el texto de la actividad.

3.- Formar la cabecera del buffer de envío al servidor con la transacción correspondiente, y además, colocar en dicho buffer los registros de actividades suficientes para llenar el buffer.

4.- Enviar el buffer al servidor y esperar respuesta.

5.- Obtener el código de retorno del servidor y mostrar el mensaje correspondiente al valor retornado.

6.- Si se tienen más registros de actividades por enviarse al servidor, repetimos los pasos a partir del 3.

7.- Una vez que no se tengan más registros que enviarse, se termina la transacción.

4.3.8. Algoritmo CON_ACT.

Este algoritmo explica cómo se realiza una consulta de actividades de la agenda por parte del cliente. La consulta de actividades se hace para un sólo dia a la vez, para la fecha que se solicite.

1.- Formar la cabecera del buffer a eviarse al servidor con la transacción correspondiente a la consulta de actividades, además enviar en el buffer la fecha para la cual se van a consultar las actividades.

2.- Enviar el buffer y esperar por la respuesta.

3.- Una vez que llegue el buffer desde el servidor, obtener de éste el código de retorno y dependiendo de su valor, mostrar el mensaje correspondiente.

4.- Del buffer que llega desde el servidor, obtener los registros correspondientes a las actividades de la fecha que se requirió. Si el servidor envió alguna seña indicando que existen más actividades para la fecha requerida, se repiten los pasos a partir del 1., colocando además en el buffer de envío al servidor, el último registro retornado por el servidor.

5.- Una vez que se tengan todos los registros de actividades, éstos son mostrados al usuario, si es que para la fecha que se requiera existen actividades.

4.3.9. Algoritmo MOD_ACT.

Este algoritmo explica como la aplicación cliente realiza la modificación de alguna actividad para una fecha dada en la agenda.

1.- Realizar una transacción de consulta de actividades para la fecha que se requiera, ésto ya se explicó en el algoritmo anterior.

2.- Una vez que se tengan los registros de las actividades que retornó el servidor para la fecha requerida, por cada actividad que se quiera modificar, elegir la hora y modificar el texto de la actividad.

3.- Una vez que se confirme la modificación de las actividades en las horas de la fecha elejida, formar la cebecera del buffer de envío al servidor con la transacción correspondiente y colocar en el buffer los registros de actividades suficientes para llenar el buffer.

4.- Enviar el buffer al servidor y esperar la respuesta.

5.- Una vez que se reciba el buffer de respuesta, obtener el código de retorno y de acuerdo al valor que tenga mostrar el mensaje respectivo. Si existen más registros de actividades por enviarse, repetir los pasos a partir 3 (con excepción de la confirmación).

6.- Una vez que no existan más registros por enviarse, se termina la transacción.

4.3.10. Algoritmo ELI_ACT.

Este algoritmo explica cómo la aplicación cliente puede eliminar actividades para una fecha dada de la agenda.

1.- Efectuar una transacción de consulta de actividades para la fecha que se requiere.

2.- Una vez que se tengan los registros de las actividades que retornó el servidor para la fecha requerida, por cada actividad que se quiera eliminar, elegir la hora y borrar el texto de la actividad.

3.- Una vez que se confirme la eliminación de las actividades en las horas de la fecha elejida, formar la cebecera del buffer de envio al servidor con la transacción correspondiente y colocar en el buffer los registros de actividades suficientes para llenar el buffer.

4.- Enviar el buffer al servidor y esperar la respuesta.

5.- Una vez que se reciba el buffer de respuesta, obtener el código de retorno y de acuerdo al valor que tenga mostrar el mensaje respectivo. Si existen más

registros de actividades por enviarse, repetir los pasos a partir 3 (con excepción de la confirmación).

6.- Si no existen más registros para eliminación, por enviarse al servidor, se termina la transacción.

4.3.11. Algoritmo PERMISOS.

Este algoritmo explica cómo un usuario, a través de la aplicación cliente, puede otorgarle permisos de lectura y quizá escritura a otros usuarios sobre su agenda.

1.- Consultar los usuarios que se encuentran en el archivo de usuarios autorizados que se encuentra en el servidor.

2.- Una vez que se tenga la lista de los usuarios, se debe elegir uno de ellos, a quien se le van a otorgar los permisos.

3.- Ingresarle al usuario los permisos, ya sea de sólo lectura o de escritura.

4.- Formar el buffer a enviar al servidor con la transacción correspondiente, y formatear los datos ingresados en el buffer.

5.- Una vez que se confirme la operación, enviar el buffer al servidor y esperar la respuesta.

6.- Del buffer de respuesta del servidor, obtener el código de retorno y de acuerdo al valor de éste, mostrar el mensaje respectivo al usuario.

4.3.12. Algoritmo CON_AGENDA.

Este algoritmo le permite a un usuario, a través de la aplicación cliente, consultar las agendas sobre las cuales él tiene permisos de sólo lectura o escritura.

1.- Formar el buffer de envio al servidor con la transacción correspondiente.

2.- Enviar el buffer y esperar la respuesta del servidor.

3.- Colocar en un arreglo los registros correspondientes a las agendas a las cuales el usuario tiene acceso.

4.- Si el servidor le envia una seña al cliente indicando que existen más registros por enviarle al cliente, el cliente repite los pasos a partir del 1.

5.- Una vez que no existan más registros que enviarle al cliente, se termina la transacción.

4.3.13. Algoritmo CON_PERMISOS.

Este algoritmo explica cómo un usuario, a través de la aplicación cliente, puede consultar los usuarios que tienen permisos de sólo lectura o escritura sobre la agenda propiedad del usuario que está usando la aplicación.

1.- Formar el buffer con la transacción correspondiente de consulta de usuarios con permisos sobre la agenda.

2.- Enviar el buffer al servidor y esperar la respuesta.

3.- Colocar los registros de permisos en un arreglo.

4.- Si el servidor envia una seña al cliente de que existen más registros por enviarse, seguir los pasos a pertir de 1.

5.- Si no existen más registros por enviarse, termina la transacción.

4.3.14. Algoritmo ACT_BROAD.

Este algoritmo explica el proceso de añadirle alguna actividad en la agenda de varios usuarios.

1.- Efectuar una transacción de consulta de actividades para la fecha que se especifica, si no se trabaja con la fecha actual.

2.- Efectuar uan transacción de consulta de usuarios autorizados al servicio de la agenda.

3.- Editar el texto de la actividad, en la fecha y hora que se haya elegido.

4.- Elegir de entre los usuarios que se consultaron a quines se les añadirá la actividad en sus agendas.

5.- Formar la cabecera del buffer con la transacción correspondiente y formatear los datos a enviar en el buffer de envio al servidor.

6.- Enviar el buffer al servidor y esperar la respuesta.

7.- De la respuesta enviada por el servidor, obtener el código de retorno y dependiendo del valor que tenga, mostar el mensaje respectivo.

4.4 COMPROMISOS DE DISEÑO.

- El protocolo en el cual se basan las aplicaciones cliente y servidora, contiene campos que a veces son llenados sólo por el cliente o sólo por el servidor. De todas maneras para el caso del cliente, éste siempre debe llenar los campos de la cebecera del buffer que se envia al servidor y que

corresponden al protocolo definido, de manera que los datos le lleguen al servidor en el orden apropiado. Con esto se logra que el servidor trabaje sin problemas de desplazamiento de caracteres en el buffer, pero teniendo caracteres de más que se transmiten (overhead).

- El cliente es el abre y cierra la conexión con el servidor, y esto lo hace cada vez que necesita enviar una transacción al servidor. Aún cuando este mecanismo evita el tener que mantener una conexión activa durante un tiempo relativamente largo con los riesgos que esto pueda tener, esto puede producir demoras en el procesamiento de transacciones, ya que por cada transacción se debe abrir la conexión, y después cerrarla.
- La aplicación cliente tiene un archivo de configuración, en el cual se tendrán datos relacionados a la máquina en la cual corre el programa servidor, el puerto de dicha aplicación, y otros datos que se pueden revisar en el apéndice No 1. Esto hace que la mayoría de los parámetros de la aplicación sean configurables y esto evita que el usuario tenga que ingresarlos cada vez que usa la aplicación. El inconveniente surge cuando el archivo tiene datos incorrectos, lo cual es responsabilidad del usuario, o si el archivo llega a perderse accidentalmente.

4.5 DISEÑO DE LAS INTERFASES GRAFICAS.

La aplicación cliente usará una interfase gráfica basada en Windows. El producto que nos permite hacer este diseño es Visual Basic. A través de Visual Basic,

creamos unos objetos que denominamos **FORMAS**, dichas formas pueden contener objetos adicionales, tales como botones, cajas de texto, temporizadores, listas, grids, listas de opción, listas de verificación, etc.

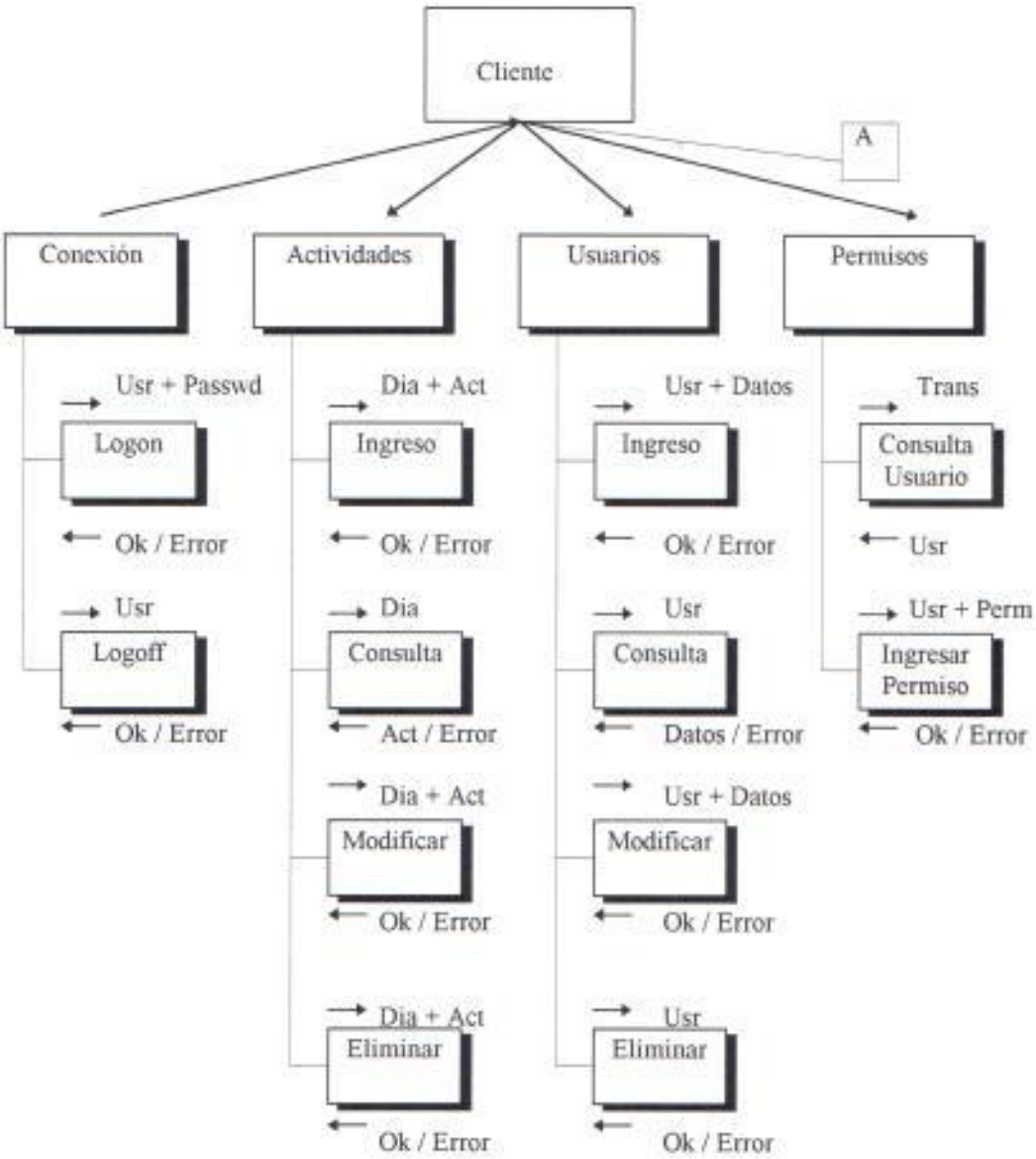
Cada objeto dentro de una forma está conectado a algún evento propio de él, cuando ocurre dicho evento y si el evento está enlazado a algún código, este código se ejecuta, la ejecución del código relacionado a un evento puede determinar ya sea la aparición de otra forma, la terminación del programa, o alguna otra operación pertinente.

Existe además una forma principal, la cual es la primera que se carga cuando el programa corre, es en esta forma en la que se hace el diseño del menú. El menú presentado en esta forma es idéntico al usado en las ventanas standar de Windows,o sea de tipo barra pop-down, con la posibilidad de contar con submenús si estos son necesarios. Para poder enlazar nuestra aplicación en Windows con las librerías que nos permiten el manejo de sockets, utilizamos objetos definidos para Visual Basic que hacen esta función.

El diseño de la interfase lo hacemos a través de la barra de herramientas de Visual Basic, en la forma que se muestra podemos ir añadiendo los objetos mostrados en la barra. Los objetos que podemos añadir pueden ser ya sea cajas de edición, listas, grids, botones, etc. Si quisiéramos añadir un menú a la forma, lo hacemos a través de la opción Windows, del menú principal. Si tenemos los controles para manejo de sockets previstos, los podemos incluir en la forma, cuando el programa corra, los iconos de ésta desaparecen.

Las ventanas a través de las cuales el usuario podrá interactuar con el programa cliente, son en formato Windows, con todas las facilidades que éstas proveen; los mensajes que se muestren al usuario, se lo hará también a través de ventanas de mensajes con un formato apropiado para el efecto.

4.6. DIAGRAMA DE BLOQUES DE LOS PROCEDIMIENTOS DEFINIDOS EN EL CLIENTE.



En los procedimientos definidos bajo : Conexión, Actividades, Usuarios, Permisos, Agendas. Se efectúa el envio de los requerimientos al servidor, y se espera por la respuesta que le envie este al cliente, esto se hace asi porque hay que recordar que por cada transacción que efectúa el cliente con el servidor, se abre y se cierra la conexión.