CAPITULO CUATRO

4 AGENTE DE USUARIO

La funcionalidad del agente de usuario en BIDACI es llevar un modelo del usuario, representando sus intereses, hábitos y preferencias, respecto al material y temas de interés de BIDACI. Cuando un usuario tiene un número determinado de visitas a diversos títulos de un tema, el agente de usuario le pregunta al usuario (ver figura 4.1), si el tema es de su interés. En base a ello el agente propone la configuración o formación de grupos de discusión automáticamente por áreas de interés. Además permite la concertación de citas, basándose en el modelo del usuario. Todo esto hace que BIDACI sea un ambiente adaptivo.

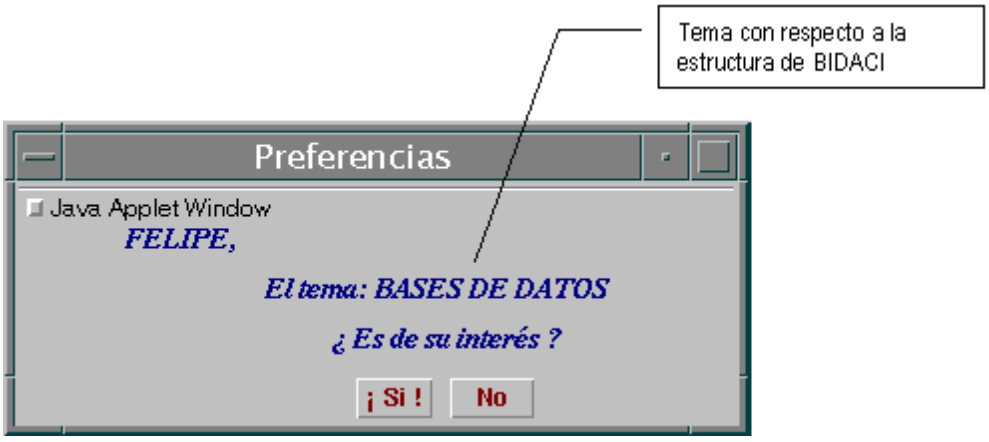

Figura 4.1 Preguntar sus preferencias al usuario.

Los criterios para que el agente de usuario permita formar un grupo de discusión son:

a). *Que por lo menos haya uno o dos participantes además del usuario*. el agente debe ayudar indicando si existen una o más personas además del usuario que presenten interés en ese tema.

✓ *Que hayan presentado interés en material del acervo referente al tema*. el grupo de discusión debe estar relacionado con alguna página web visitada.

Para el nombre del grupo de discusión el agente elige de la lista de temas de acuerdo a la taxonomía de BIDACI – ANIEI, Para formar un grupo de discusión el agente permite visualizar los posibles integrantes para ese grupo (ver figura 4.2).

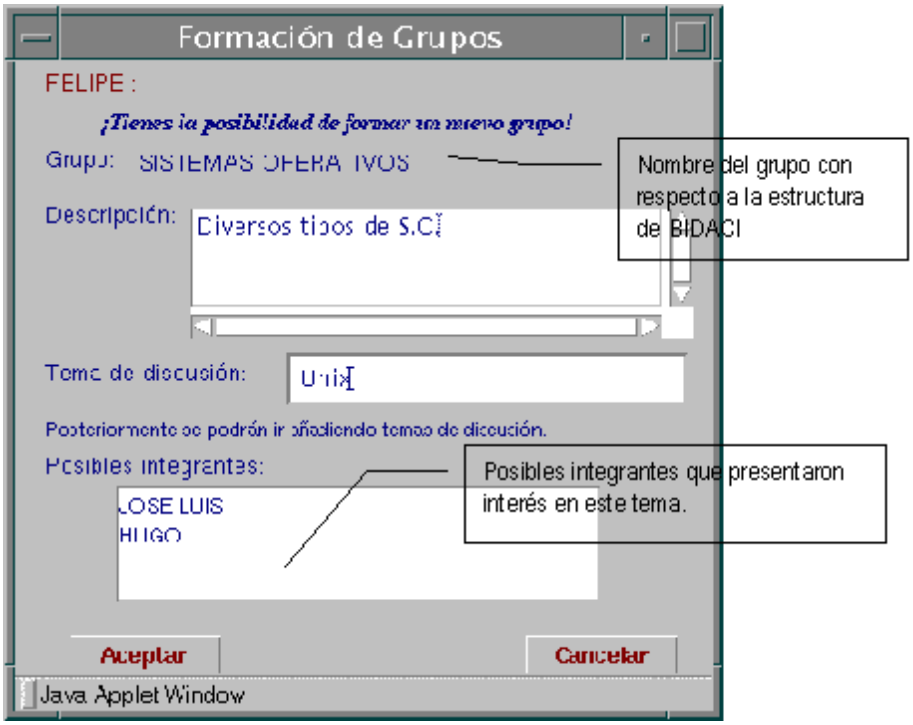

Figura 4.2. Formación de un grupo de discusión.

El agente notificará a los usuarios indicados como posible participantes si desean ser incluidos en el grupo de discusión. El agente se encargará de enviar un correo electrónico informando a cada uno que entren al ambiente para indicar si desean ser miembros o formar parte del grupo de discusión.

El agente tendrá la capacidad de eliminar un grupo de discusión en el momento en que quede un solo miembro (usuario creador del grupo) al hacerlo, enviará un correo electrónico de aviso a la última persona o miembro que haya quedado.

Las restricciones para el agente de usuario no permita crear los grupos de discusión son:

- Que nadie presente interés en el tema.
- Que no exista material en el acervo de ese tema.

Una vez formados los grupos de discusión, es necesario convocar citas para una sesión en el salón virtual (ver figura 4.3). El agente se encargará de hacer llegar las invitaciones a los participantes previamente elegidos por el usuario (ver figura 4.4). Si se confirma que una o más personas participarían en la sesión, notificará al que haya concertado la cita que ya se formalizó la sesión, para la fecha y hora correspondiente. En caso de que ningún participante desee estar en la salón virtual, el agente de usuario se encargará de enviar un correo electrónico que dicha sesión fue cancelada. El agente podrá enviar correos electrónicos el día de la sesión 3 horas antes para recordar a los participantes de dicha sesión.

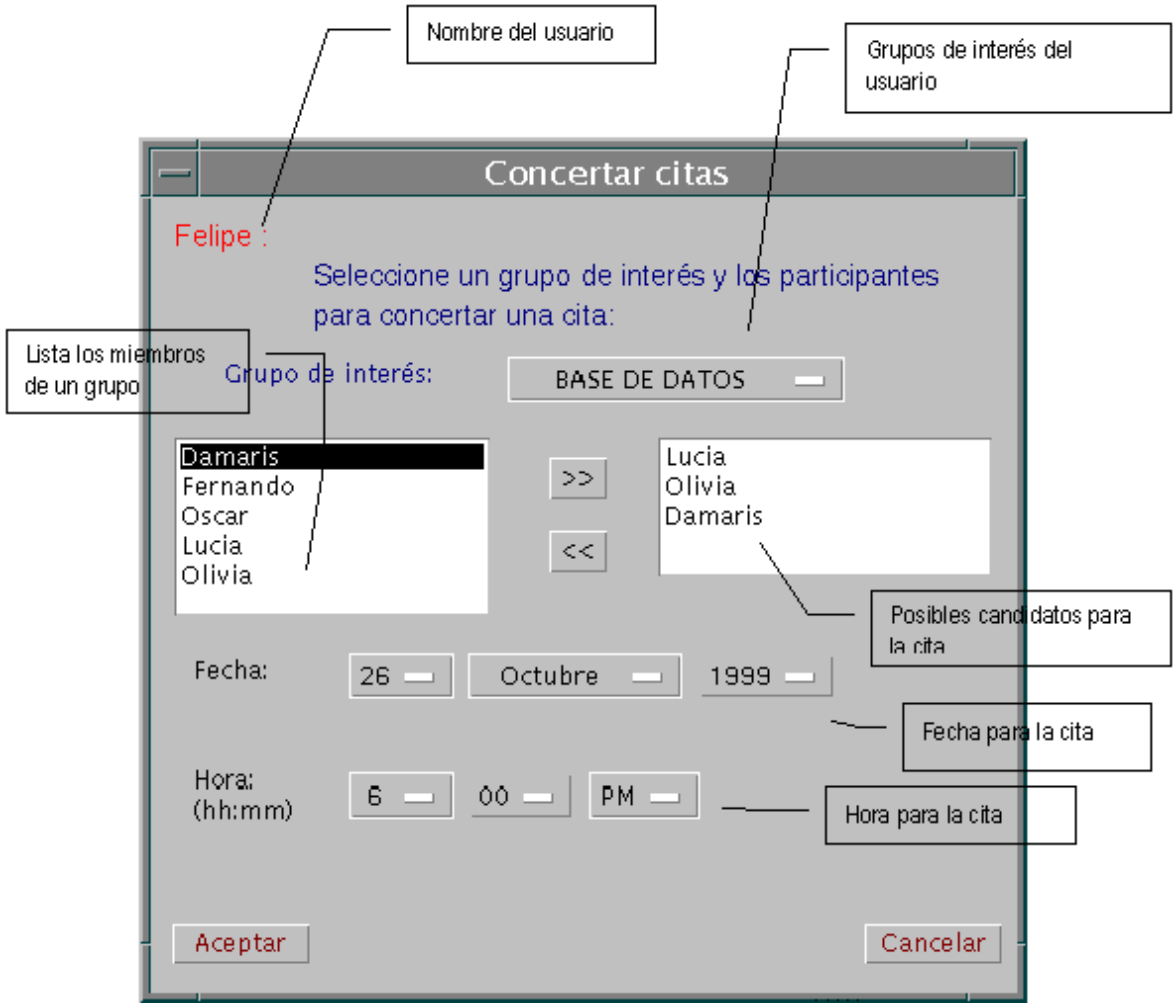

Figura 4.3. Concertación de citas.

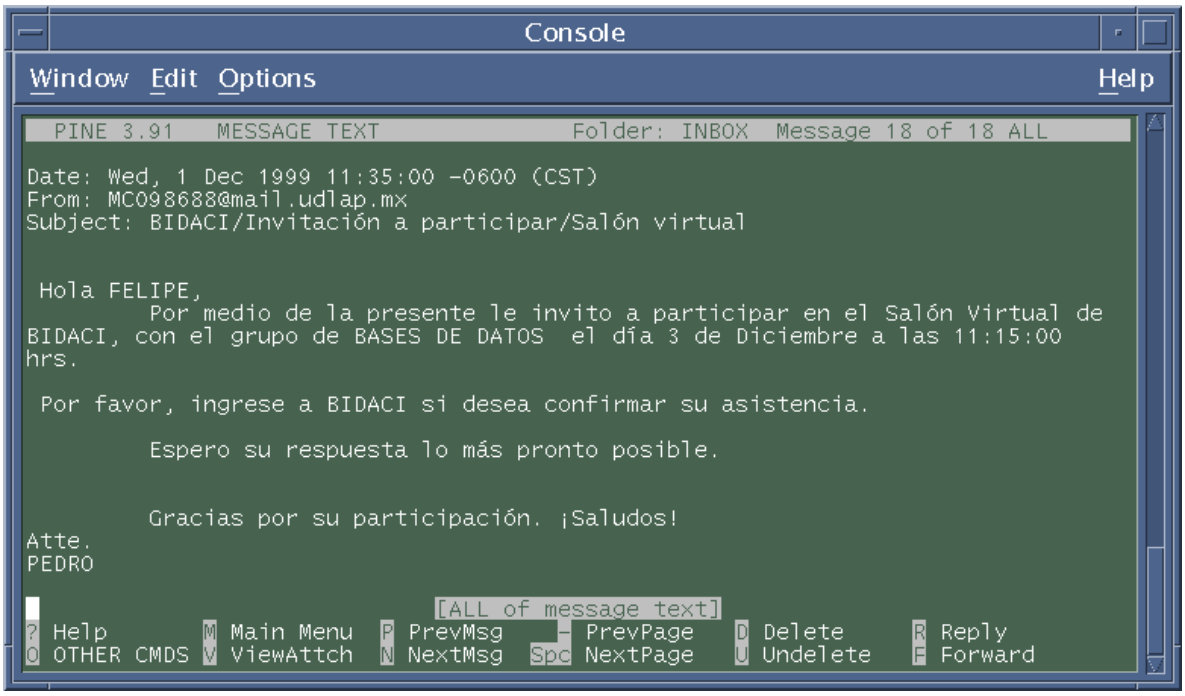

Figura 4.4 Invitación para el salón virtual.

Para el envío de correos electrónicos se implemento un servidor de socket (*ServerSocket*) y un socket, para evitar los problemas de seguridad de Java en el navegador. El servidor registra el *número de puerto* disponible. El agente de usuario y el agente administrador (Clientes) pedirá conectarse con el servidor por este *puerto*.

El *ServerSocket* (socket servidor) establece el puerto en el que el servidor esperará las conexiones de los clientes. Cada conexión se maneja con un *Socket* (Cliente) [Deitel & Deitel,98]. El siguiente código que se muestra la interacción Cliente/Servidor con conexiones de sockets de flujos:

```
// Código fuente: Servidor.java
// Recibe la conexión de Cliente
// y cierra la conexión.

import java.io.*;
import java.net.*;
import java.lang.*;

public class Server {

   public static void main (String args[])
   {
     //Declaraciones
    ServerSocket s= null;
    Socket s1;

    InputStream s1in;
    DataInputStream dis;

     //Atributos
    String Desde,Para,Subject,Mensaje;
    EnviarUnEmail envia;


        //Creación de un nuevo servidor con número de puerto 5432
        //y número de clientes que pueden esperar una conexión y
        //ser procesador por el servidor.
    try{
        s = new ServerSocket( 5432, 100 );
    }catch(IOException e){ System.out.println("Creación :" +e);
    }

        // El servidor escucha indefinidamente
    while(true){
     try{
          //Este método devuelve un objeto socket
          //cuando se establece la conexión.
        s1 = s.accept(),
         //Se invoca al método getInputStream para el socket
```

```
          //a fin de obtener una referencia al InputStream
          //asociado al Socket
         s1in = s1.getInputStream();
         dis = new DataInputStream(s1in);

           //El método readUTF permite leer la cadena
         Desde = new String(dis.readUTF());
         Para = new String(dis.readUTF());
         Subject = new String(dis.readUTF());
         Mensaje =  new String(dis.readUTF());

         //Se crea un objeto de tipo EnviarUnEmail
         //para enviar el email desde el servidor
         envia = new EnviarUnEmail();
         envia.send(Desde,Para,Subject,Mensaje);



       // Cierra la conexión
          dis.close();
          s1in.close();
          s1.close();

      }
      catch ( IOException e ) {
         System.out.println("Flujo :" +e);
      }
    } //fin del while
  } //fin del main

}
```

El código del cliente se muestra a continuación:

```
/**************************************************************************
```

Clase EmailCliente:

Esta clase permite conectarse al servidor y enviarles los datos para el email.

Titulo: "Modelado de Agentes para una Biblioteca Digital en Informática"
Autor: José Felipe Cocón Juárez
Asesor: Dr. Gerardo Ayala San Martín
Versión: 1.0

Creación: 19-Oct-99
Ciclo Escolar: Otoño '99

Maestría en Ciencias con especialidad en Ingeniería en Sistemas Computacionales
Fundación Universidad de las Américas Puebla

**********/

```
import java.awt.*;
import java.applet.*;
import java.net.*;
import java.io.*;


public class EmailCliente {

  public EmailCliente (URL base, String Desde, String, Para, String Subject, String
Mensaje)
  {
   Socket s1;

   InputStream s1out;
   DataInputStream dos;

   /* Con el método getHost Obtengo el host del URL */
   try{
       s1 = new Socket(base.getHost(), 5432);
       slout = s1.getOutputStream();
       dos = new DataInputStream(s1out);

        dos.writeUTF (Desde);
        dos.writeUTF(Para);
        dos.writeUTF (Subject);
        dos.writeUTF (Mensaje);

        dos.close();
        s1out.close();
        s1.close();

    }
    catch ( IOException e ) {
      System.out.println("Flujo :" +e);
    }
  } //fin del cliente
```

}//fin de class

El siguiente código permite realizar las conexiones al servidor de correos:

```
/**************************************************************************
Clase EnviarUnEmail:

        Esta clase permite enviar Email en forma automatica a cualquier
        academico y/o estudiante que pertenezca al ambiente.



Titulo: "Modelado de Agentes para una Biblioteca Digital en Informática"
Autor: José Felipe Cocón Juárez
Asesor:         Dr. Gerardo Ayala San Martín
Versión: 1.0
Creación: 19-Oct-99
Ciclo Escolar: Otoño '99

Maestría en Ciencias con especialidad en Ingeniería en Sistemas Computacionales
Fundación Universidad de las Américas Puebla



**************************************************************************/

 import java.net.*;
 import java.io.*;
 import java.util.*;

 public class EnviarUnEmail
 {

 //métodos


        void send(String admin,String Email,String subject,String mensaje)
        {

                Socket s1 = null;
                DataInputStream sIn1 = null;
                DataOutputStream sOut1 = null;
                // open connection on port #25
```

```
 try
 {    // servidor de correos electrónicos, puerto 25
   s1 = new Socket("mailweb.pue.udlap.mx",25);
   sIn1 = new DataInputStream(s1.getInputStream());
   sOut1 = new DataOutputStream(s1.getOutputStream());
 }
 catch (UnknownHostException e)
 {
   System.out.println("Servidor desconocido.. 1 ??" +e);
 }
catch (IOException e)
{
   System.out.println("Conect..??" +e);
}

if(s1!=null && sIn1!=null && sOut1!=null)
{
   try
    {
     sOut1.writeBytes("MAIL From: "+admin+" \n");
     sOut1.writeBytes("RCPT To: "+Email+"\n");
     sOut1.writeBytes("Data\n");
     sOut1.writeBytes("From: "+admin+" \n");
     sOut1.writeBytes("Subject: "+subject+" \n");
     sOut1.writeBytes(mensaje);
     sOut1.writeBytes("\n.\n");

     String respuesta="";

     while ((respuesta = sIn1.readLine())!=null)
     {
          System.out.println("Desde servidor: "+ respuesta);
          break;
     }

     sOut1.close();//Cerramos todo lo que hemos abierto
     sIn1.close();
     s1.close();
    }
    catch (UnknownHostException e)
    {
     System.out.println("Servidor desconocido..11 ?? "+e);
    }
  catch (IOException e)
  {
   System.out.println("Error de conexion " + e);
  }
```

```
                } // fin del if

        } // fin del send
}// Fin de la Clase.
```

4.1 Modelo del Agente de Usuario

En la siguiente tabla se presenta el conjunto de clases y métodos que conforman el modelo del agente de usuario:

Clases	**Atributos**	**Métodos**
UserAgent	Username_usuario, nombre_usuario, tema_usuario, titulo_usuario, url_usuario, numero_accesos_preferencias, fecha_preferencias. Preferencias, Grupo.	obtenerTema, verificarModelo, actualizaPreferencias, notificarLaCita, verificarPosibleMiembroGrupo, depurarCitas, depurarGrupo.
Preferencias	Tema_usuario, nombre_usuario, username_usuario.	presentarElTemaDePosibleInterés, eliminaPrefencias.
Grupo	Username_usuario, tema_usuario, posibles_participantes, email_participantes	verificaGrupoExistente, presentarGrupoPosiblesMiembros, creaGrupoNuevo, AgregaUsuarioAlGrupo, EnvíaEmailDeInvitación
SesiónChat	Username_usuario, nombre_usuario, grupos_usuario,	presentarLosGruposDelUsuario, presentarLosMiembrosDelGrupo, preguntarSiEnviaLosEmails,

	usuario_miembros posibles_candidatos, fecha_cita, hora_cita. EmaiCliente.	darDeAltaLaCita,
EmailCliente	Desde, para, titulo, mensaje.	enviaDatosAlServerSocket.

Tabla 4.1 Estructura del agente de usuario

El papel que juega el agente de usuario en BIDACI es cumplir ciertas tareas específicas en la cual fue programado, como el de envío de correos electrónicos para la concertación de citas para el salón virtual, la formación de grupos con un número de accesos al tema de interés y el de preguntar si los temas son su preferencia.

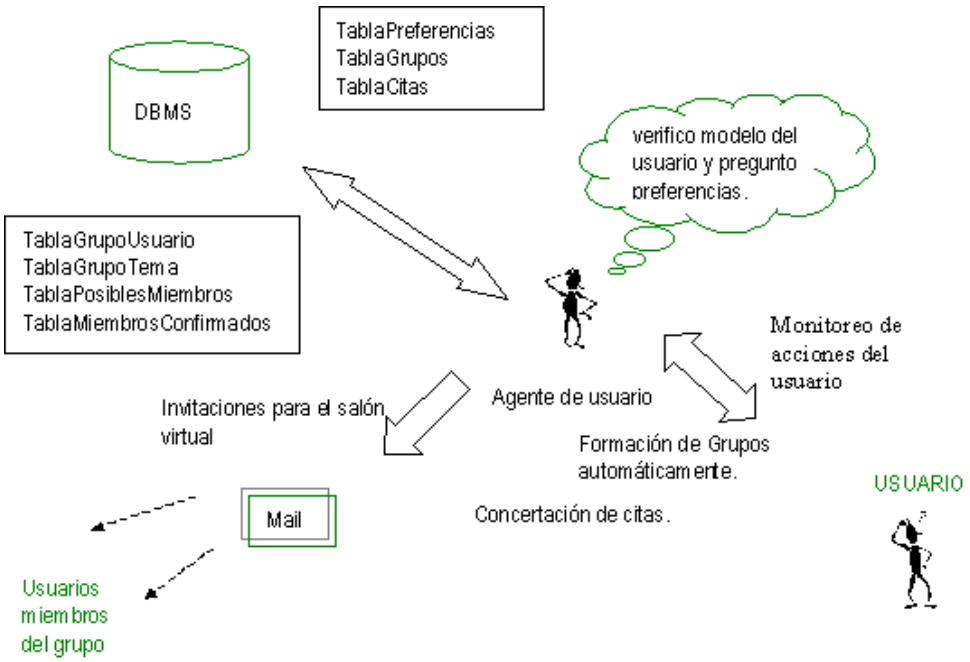

Figura 4.5 Arquitectura del agente de usuario.

En la arquitectura mostrado en la figura 4.5 en donde se logra la participación del agente de usuario con el usuario, así como sus acciones para el cual fue diseñado.

En la siguiente figura se muestra el modelo entidad-relación de las tablas utilizadas en el DBMS para el desarrollo del agente de usuario, ya que las demás tablas fueron definidas por Agosto [1998].

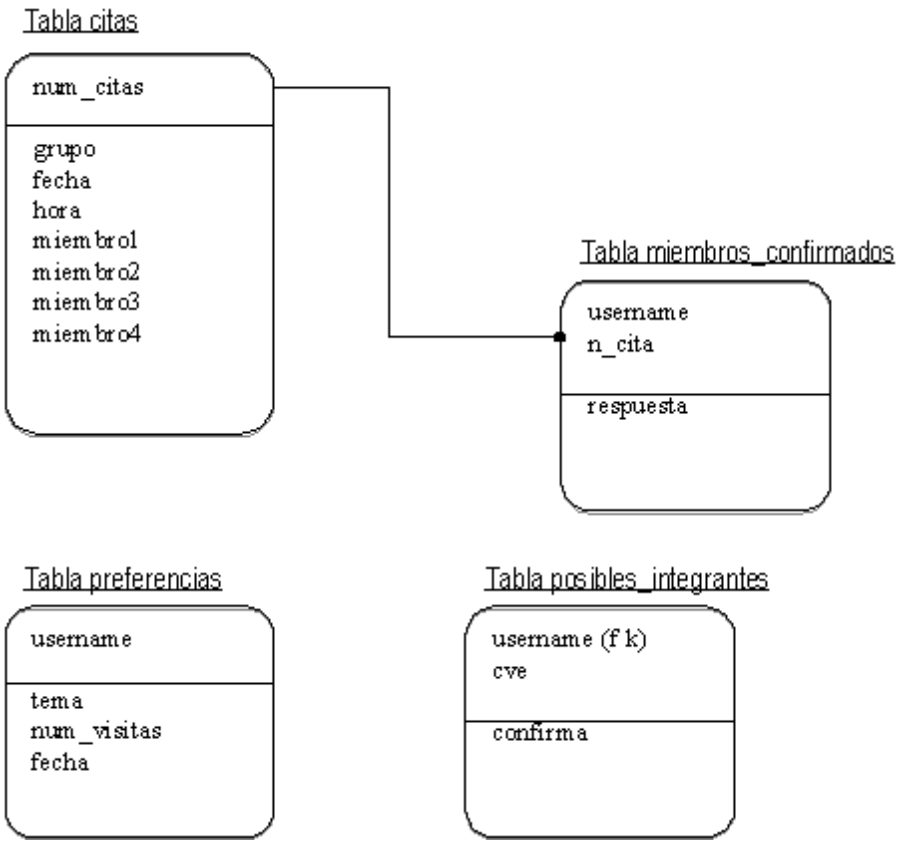

Figura 4.6 Estructura de la tablas implementadas para el agente de usuario.

4.2 DESCRIPCIÓN DEL MODELO DEL AGENTE DE USUARIO.

Clase implementada:

UserAgent

Esta clase es subclase de la clase Frame y contiene los siguientes atributos:

username_usuario,

nombre_usuario,

tema_usuario,

titulo_usuario,

url_usuario,

numero_accesos_preferencias,

fecha_preferencias.

Los métodos implementados son:

obtenerTema.

Este método obtiene el tema de la base de datos y lo asigna al atributo tema_usuario basándose en las preferencias del usuario.

verificarModelo.

Este método verifica el modelo para cada usuario y revisa los números de accesos al tema de interés del usuario.

insertarNuevoUsuarioEnPreferencias.

Este método inserta un nuevo usuario en la tabla preferencias, en caso de que el usuario no exista en la tabla preferencias.

actualizaPreferencias.

Este método es utilizado para actualizar las preferencias de cada usuario.

notificarLaCita.

Este método permite notificar a los usuarios las posibles citas para el salón virtual en donde han sido invitados.

verificarPosibleMiembroGrupo.

Este método permite notificar a los usuarios que son posibles integrantes para un grupo y confirma su asistencia.

depurarCitas.

Este método mantiene las citas actuales, ya que las citas pasadas las elimina basándose en la fecha y hora de la sesión, de igual manera elimina las concertaciones de citas que no se hayan formalizado, notificando al miembro que la sesión ha sido cancelada por falta de quórum.

depurarGrupo.

Este método elimina los grupos que solo contengan un usuario basándose en las respuestas de los posibles integrantes para ese grupo y también utiliza la clase EmailCliente, para notificar que el grupo fue borrado porque no existen miembros para ese grupo.

PreguntarPreferencias

Esta clase es subclase de la clase Dialog y los atributos que contiene:

tema_usuario,

nombre_usuario,

username_usuario.

Los métodos que utiliza son:

presentarElTemaDePosibleInterés.

Este método utiliza el objeto varTema de tipo Label, para presentar el tema de interés para el usuario.

eliminaPrefrencias.

Este método elimina el tema de interés del usuario dada su respuesta negativa hacia ese tema.

Los principales objetos utilizados en esta clase:

agenteusuario.

Este objeto es una instancia de la clase UserAgent, en donde obtenemos el nombre del usuario y el username del usuario, los cuales son asignados a los atributos nombre_usuario y username_usuario.

varNombre.

Este objeto es de tipo Label en donde es mostrado en el Dialog

varTema.

Este objeto es de tipo Label en donde es mostrado en el Dialog

botonSi.

Este objeto es de tipo Button en donde nos permite tomar la respuesta del usuario desde el Dialog

botonNo.

Este objeto es de tipo Button en donde nos permite tomar la respuesta del usuario desde el Dialog

FormaciónGrupos

Esta clase es subclase de la clase Frame y sus atributos son:

username_usuario,

tema_usuario,

posibles_participantes,

email_participantes

Los métodos utilizados en esta clase son:

verificaGrupoExistente.

Este método verifica si existe el grupo de interés para el usuario, en caso de no existir el grupo utiliza el método creaGrupoNuevo.

presentarGrupoPosiblesMiembros.

En este método se presentan los posibles integrantes del grupo con el mismo tema de interés del estudiante.

creaGrupoNuevo.

Este método crea un grupo nuevo de discusión.

AgregaUsuarioAlGrupo.

Este método agrega un usuario al grupo de discusión existente.

SesionChat.

Esta clase es subclase de la clase Frame y sus atributos son:
username_usuario,
nombre_usuario
grupos_usuario,
usuarios_miembros,
usuarios_candidatos,
fecha_cita,
hora_cita

Los métodos utilizados en esta clase son:

presentarLosGruposDelUsuario.

Este método presenta los grupos existentes en donde se encuentra el usuario.

presentarLosMiembrosDelGrupo.

Este método presenta todos los miembros de cada grupo elegido por el usuario.

preguntarSiEnviarLosEmails.

Este método notifica al usuario, si esta seguro de enviar los emails para concertar la cita e invoca a la clase EmailCliente.

darDeAltaLaCita.

Este método inserta en la base de datos los datos de la cita, el nombre del grupo, fecha, hora y los miembros convocados.

EmailCliente.

Esta clase realiza la conexión al servidor de socket desde el Applet y sus atributos son:

Desde,

Para,

Titulo,

Mensaje

El principal método utilizado en esta clase es:

envíaDatosAlServerSocket.

Este método realiza el envío de los atributos al servidor socket.