Fco. Javier Martín Navarro

TFC-J2EE

12-1-15

Trivial OnLine

Índice

Contexto del proyecto

Un poco de historia

Parece que no ha pasado el tiempo desde que aquellos estudiantes desarrollaran en 1979 el primer juego online.

Dungeons & Dragons era la primera posibilidad de jugar a través de la red mediante una interfaz de texto mediante la cual, los usuarios podían ir avanzando en la historia.

Más de treinta años después, somos capaces de gestionar todo un ejercito de personas reales que avanzan estratégicamente para capturar un pueblo defendido fuertemente por otras decenas de usuarios, y todo ello con los gráficos más espectaculares que se han podido ver jamás.

Era entonces impensable la calidad de los títulos que se barajan hoy en día en el mundo de los juegos online y, sin embargo, han sido grandes cantidades de dinero lo que se ha invertido en esta industria al observar la productividad y el mercado que existe en la misma. De ahí la importancia que ha adquirido últimamente empresas como Kongregate, cuya función ha sido la de proveer el trabajo de otros desarrolladores por medio de un portal de juegos que permite al usuario encontrar todo lo que desee totalmente organizado por categorías.

Esta empresa nació en el 2006 con la intención de convertirse en todo un referente de los juegos online. El catálogo de posibilidades del que disponía les llevó a aumentar su capital en el 2008 a una cifra de 9 millones de dólares, y todo gracias a la cantidad de inversores que aprovecharon el momento para disfrutar de los beneficios de este sector.

Finalmente, en el 2010, GameStop decidió comprarlo, impulsando finalmente a todos los desarrolladores que habían iniciado sus relaciones con esta empresa para aumentar la oferta actual. Hoy en día, cuentan con más de 62.000 títulos, lo que los convierte en el referente que querían ser en los comienzos del portal.

El proyecto

Este proyecto esta basado en la realización de un juego on-line, realizado íntegramente en el lenguaje de programación Java.

Este desarrollo, utiliza la base de los conocidos juegos donde se realiza una pregunta y existen varias respuestas posibles y tan solo una de ellas es la correcta.

Para poder participar, el jugador debe conectarse a una pagina web -desde la cual se accede al applet Java, dandole los parámetros iniciales para arrancar- y registrarse previamente en el sistema, donde queda almacenado su nombre de usuario, contraseña, fecha y la IP desde la cual se ha conectado.

Tras identificarse correctamente en el sistema, accede a la sala de juego y podrá observar todos los participantes que están en juego y las puntuaciones de los máximos acertantes. Así mismo, podrá comenzar a participar, respondiendo a las preguntas y sumando puntos, si acierta las respuestas.

En la base de datos se recoge toda la información del usuario, desde sus datos ya mencionados anteriormente, hasta sus puntuaciones. Este sistema solo podrá ser visualizado por el gestor del servidor, ya que se almacenara con una contraseña que evitara que usuarios malintencionados puedan acceder a los datos de los participantes.

El almacenamiento de datos puede estar ubicado en la misma carpeta que el programa o en otra diferente ya que esta dirección dentro del servidor se puede especificar en la programación del juego y permite que, en caso de ataques al sistema, sea mas difícil localizar los datos de los usuarios.

Del mismo modo, el sistema gestor de la base de datos, trabaja con un puerto -a la escucha de peticiones de conexión- determinado, que tendrá que ser abierto y re-direccionado en el router de salida a internet para poder trabajar correctamente con el programa.

La base de datos utiliza el lenguaje SQL como medio de comunicación con Java y estará basado en algún software lo mas abierto y accesible posible, como puede ser MySQL o Access.

En la programación del juego, se tienen en cuenta caracteres específicos, como por ejemplo el uso de comillas, puntos, comas, paréntesis, interrogantes, etc de tal forma que, si son utilizados en el registro del usuario, sean automáticamente denegados y evitar así, ataques de inyección SQL.

Caso de uso

Juan es un alumno de Bachiller con muchas ganas de empezar en la universidad. Para ello esta barajando diferentes opciones y la que mas le gusta, al menos de momento, es la UOC.

Hoy ha quedado con unos amigos mas de su instituto porque ha visto un juego dentro de la página de la universidad que le ha llamado la atención, y es que pueden conocer la universidad y todo lo que ofrece pasando un rato divertido. Además, este juego permite que cada uno desde su casa se conecte y puedan competir entre ellos para saber mas sobre lo que quieren estudiar, como aprenderlo, como es estudiar a distancia y sobre todo, como no voy a inscribirme en esta universidad si ya solo entrar estoy jugando y aprendiendo.

Juan y sus amigos se conectan a la pagina www.uoc.edu, allí hacen clic sobre el enlace "Conoce tu universidad jugando" y les pide a cada uno de ellos que elijan un nombre de usuario y una contraseña.

Ya están todos registrados, menos Carlos, que dice a Juan que no le deja introducir símbolos en el nombre …. Pues no los pongas!!, le contesta Juan; ahora ya pueden ver sus nombres en una columna junto a otros participantes en el juego. Además, hay una foto sobre la universidad y están preguntando si sabemos donde esta ubicada ….. –Como no lo vamos a saber, esta en Barcelona!!!!- nos reímos mientras chateamos en la parte inferior, en una especie de Messenger donde hablan todos los participantes.

Pasa a la siguiente pregunta y ahora aparece una fotografía de lo que parece ser un aula virtual; ahora nos pregunta que donde nos dirigiríamos si necesitamos realizar una consulta sobre la matricula, Juan y sus amigos lo piensan y deciden que debe ser en Secretaria. Hemos acertado!!!, exclama en el chat el amigo de Juan.

Van pasando las preguntas y las fotografías y cada vez están mas enganchados; Juan le comenta a sus amigos que tenia dudas entre Telecomunicaciones e Informática pero que se esta decantando por esta ultima. Uno de sus amigos comenta que, una de las ultimas preguntas habla sobre las PEC y es como hacer un examen del instituto pero desde casa.

Todos quieren inscribirse en la UOC, les parece fantástico estudiar así. Es mas, han quedado al día siguiente con algunos compañeros/as de su instituto para que vean sus records en los aciertos de preguntas, les animan para que se matriculen y vean las ventajas de estudiar en la UOC.

Pedro, la persona responsable de la base de datos del juego, revisa periódicamente el sistema y comprueba que todo fnciona bien, las IPs de los concursantes, horarios, fechas, etc.

Propuesta tecnológica

El desarrollo de este proyecto esta íntegramente realizado en Java bajo la plataforma Eclipse, utilizando las librerías que incorpora y enlazando con la base de datos MySQL a través del lenguaje SQL.

Se divide en dos partes, el lado servidor y el lado cliente, distribuidos en tres clases cada uno, el juego, las constantes y el "handler" o manejador de la clase.

El desarrollo incluye las siguientes librerías:

* java.net

* java.io

* java.util

* java.sql

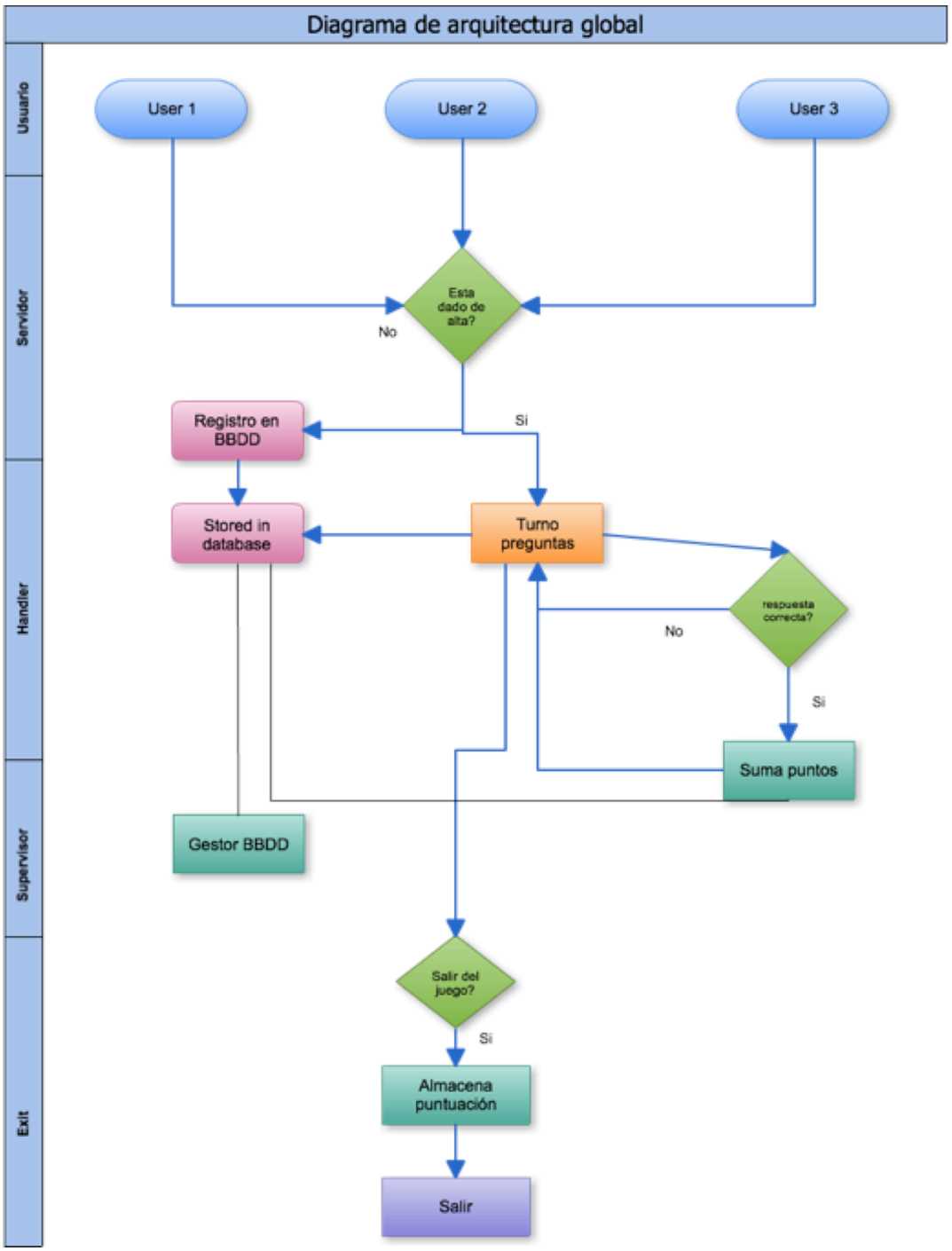
* java.text

* java.swing

* java.awt

En el siguiente apartado, se puede ver una visión global a través de un diagrama, de la configuración y relación entre las clases.

Diagramas de arquitectura



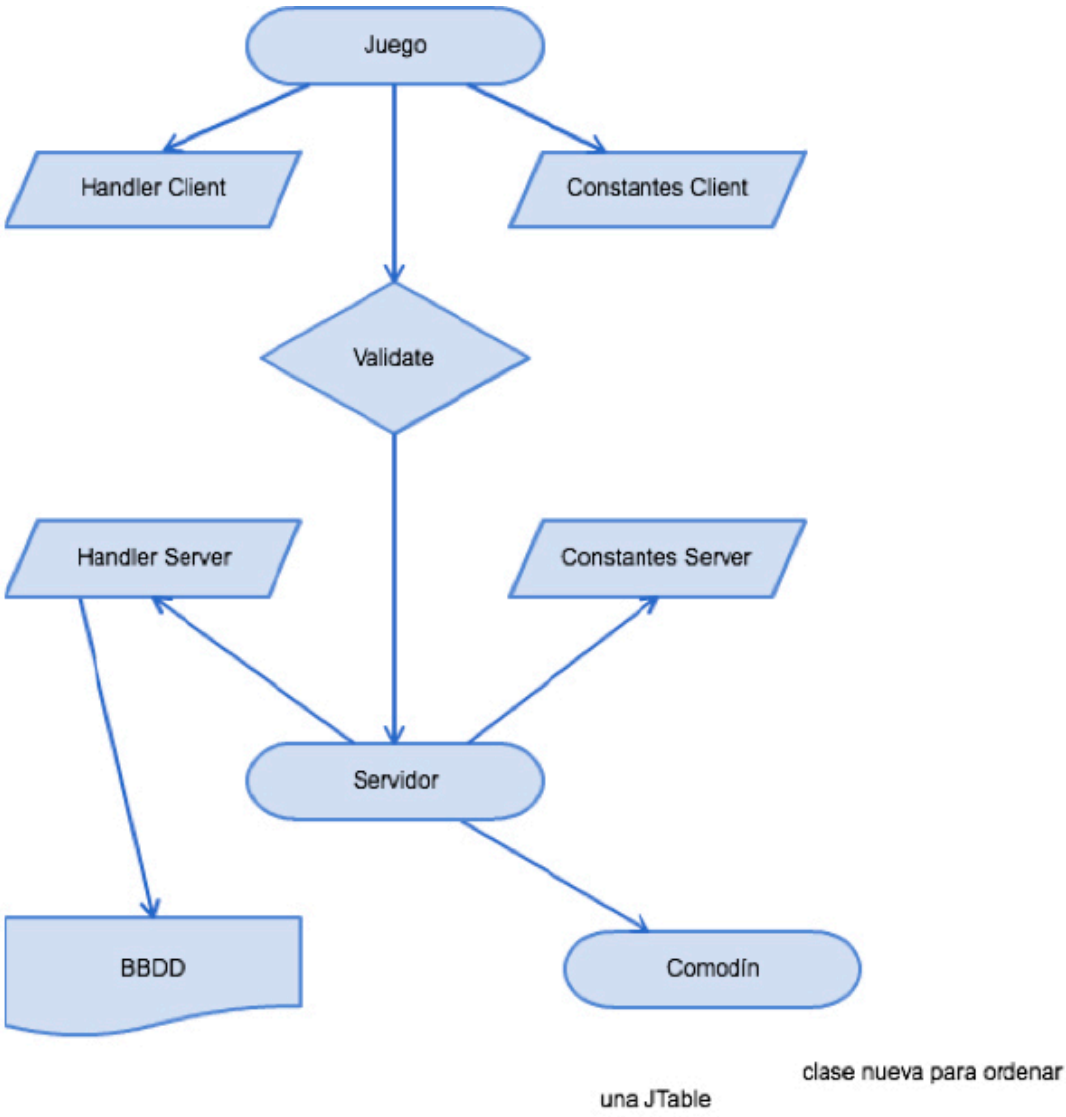
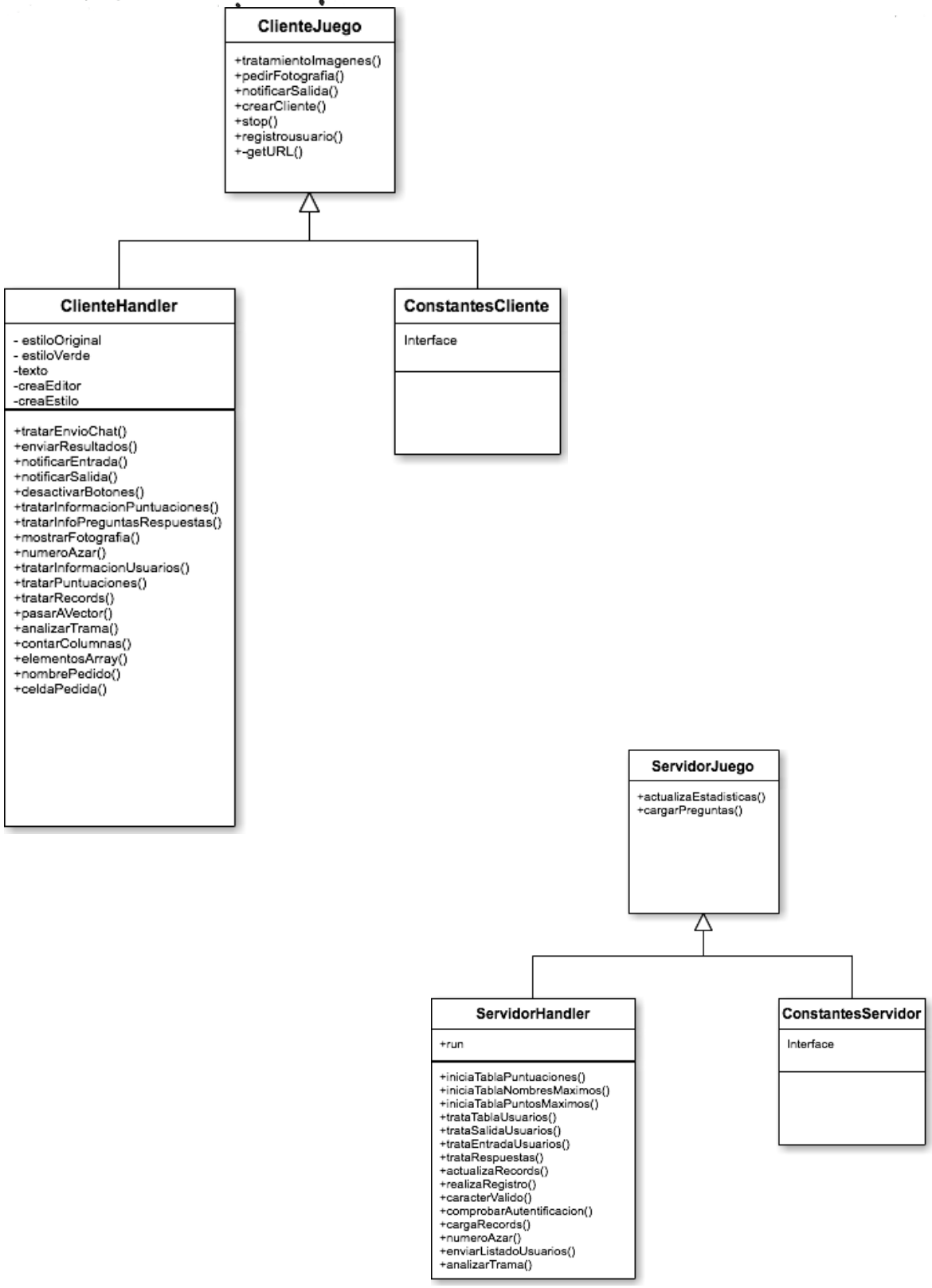


Diagrama UML


ClienteJuego
+tratamientoImagenes()
+pedirFotografia()
+notificarSalida()
+crearCliente()
+stop()
+registrousuario()
+-getURL()
ClienteHandler
- estiloOriginal
- estiloVerde
-texto
-creaEditor
-creaEstilo
+tratarEnvioChat()
+enviarResultados()
+notificarEntrada()
+notificarSalida()
+desactivarBotones()
+tratarInformacionPuntuaciones()
+tratarInfoPreguntasRespuestas()
+mostrarFotografia()
+numeroAzar()
+tratarInformacionUsuarios()
+tratarPuntuaciones()
+tratarRecords()
+pasarAVector()
+analizarTrama()
+contarColumnas()
+elementosArray()
+nombrePedido()
+celdaPedida()
ConstantesCliente
Interface
ServidorJuego
+actualizaEstadisticas()
+cargarPreguntas()
ServidorHandler
+run
+iniciaTablaPuntuaciones()
+iniciaTablaNombresMaximos()
+iniciaTablaPuntosMaximos()
+trataTablaUsuarios()
+trataSalidaUsuarios()
+trataEntradaUsuarios()
+trataRespuestas()
+actualizaRecords()
+realizaRegistro()
+caracterValido()
+comprobarAutentificacion()
+cargaRecords()
+numeroAzar()
+enviarListadoUsuarios()
+analizarTrama()
ConstantesServidor
Interface

Descripción del proyecto

El proyecto “trivial Online” tiene como principal objetivo el aprendizaje por medio del juego. Este metodo utiliza internet como via de unión entre todos los participantes y ofrece la posibilidad de jugar desde cualquier parte del mundo. A su vez, este proyecto puede resultar atractivo para conocer lugares, ofrecer servicios, examenes tipo test e incluso puede usarse como metodo medicinal con personas o niños con dificultad de relacionarse.

La aplicación se divide en dos partes muy diferenciadas, la parte del cliente y la parte del servidor.
El lado cliente, se encarga de gestionar las comunicaciones con el servidor y la presentación de la parte grafica en pantalla.

El lado servidor se encarga de registrar al usuario, cargar las preguntas, respuestas y fotografias y de gestionar el normal desarrollo del juego. Asi mismo, se encarga de relacionar a todos los usuarios a traves del chat.

Ambos lados estan divididos en varios modulos o funciones que se reparten el funcionamiento del juego. Cabe destacar los modulos de ClienteHandler, ServidorHandler y Trivial que son los que llevan el peso del juego y en su interior estan las funciones que se describen posteriormente.

Por ultimo, tenemos la Base de Datos del sistema, donde se almacenan por un lado, las preguntas y respuestas, por otro las fotografias,y por otro el registro de todos los usuarios.

Alineación con calendario de asignatura

PEC1	Desarrollo	Realización[1]	Notas
	Presentación proyecto	28/9/14	
	Propuesta	28/9/14	
	Planificación	28/9/14	

PEC2	Desarrollo	Realización	Notas
	Java - Cliente	6/10/14	Presentación de la parte gráfica del juego
	Java - Handlers	22/10/14	Programación de los Handlers del juego
	Java - Constantes	4/11/14	Programación de las constantes del juego

PEC3	Desarrollo	Realización	Notas
	Java - Server	5/11/14	Programacion del servidor del juego
	Java - Handler	20/11/14	Programación del handler del juego
	Java - Constantes	7/12/14	Programación de las constantes del juego
	BBDD	9/12/14	Programación de la base de datos
	Pruebas	19/12/14	Pruebas de funcionamiento del sistema y corrección de errores

Presentación	Desarrollo	Realización	Notas
	Memoria	9/1/15	Desarrollo de la memoria del proyecto
	Presentación	12/1/15	Ubicación del juego en un servidor para presentación del funcionamiento del juego

Descripción del funcionamiento

Esquema típico de evolución en el juego.

- Un cliente o jugador se conecta al juego para ver que puede ofrecerle.
- El sistema le solicita que se identificque o que se registre.
- Si elige registrarse, le solicitara nombre, nick, correo electronico y password, y de manera transparente para el usuario, tambien se almacena en la base de datos, la hora y la fecha asi como la IP desde donde se esta conectando.
- Si ya esta dado de alta o si ya se ha registrado, debe introducir su nick y su password.
- Al acceder a la zona de juego, debe esperar a que termine la pregunta que actualmente esta en juego, para que pueda comenzar a participar.
- Cuando comienza una nueva pregunta, tendra un tiempo determinado para responder y a la vez puede chatear con el resto de participantes, en una sala de conversacion abierta, en la parte inferior.
- El jugador puede tener hasta tres respuestas incorrectas antes de terminar su partida.
- Si el cliente desea adelantar la finalizacion de su juego en el sistema, tan solo debera cerrar la ventana web que contiene el juego.

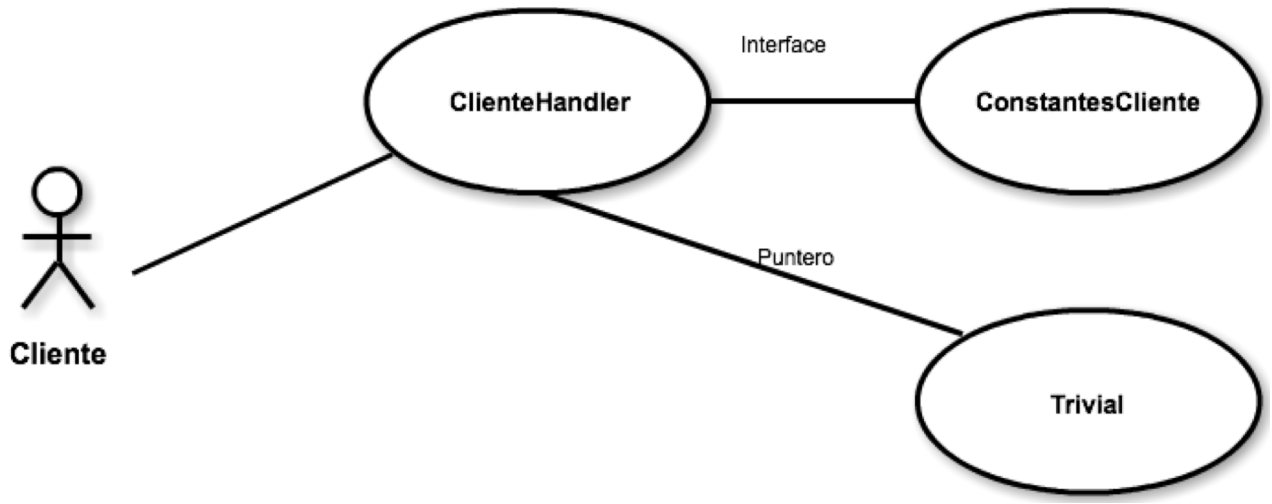
La gestión de la base de datos se lleva a cabo directamente desde la misma, sin tener un sistema gestor que desde el sistema pueda llevar un control de altas, bajas, etc... desde el propio juego.

Requerimientos funcionales

Modulo cliente

Este modulo permite la gestión de los clientes del juego. Permite la gestión de acceso al juego (conexión del socket, ubicación de los componentes en pantalla, colores y tipos de letra).

La descripción gráfica de los casos de uso es la siguiente:



A continuación se detalla la descripción textual de los diferentes casos de uso de este modulo:

Caso de uso número 1: ClienteHandler

Resumen de funcionalidad	Llama al applet Trivial.
Papel dentro del trabajo del usuario	Maneja a cada jugador dentro del juego.
Actores	Cliente.
Casos de uso relacionados	
Precondición	
Flujo normal	1- El sistema nos pide Nick y contraseña 2- Si estamos registrados, entramos a la sala de juego. 3- Esperamos a que termine la pregunta en curso. 4- Participamos en el juego.
Flujo alternativo	5- El cliente puede chatear con el resto de participantes. 5- El cliente puede abandonar la sala de juego, cerrando la pantalla del navegador.
Postcondición	
Aclaraciones	

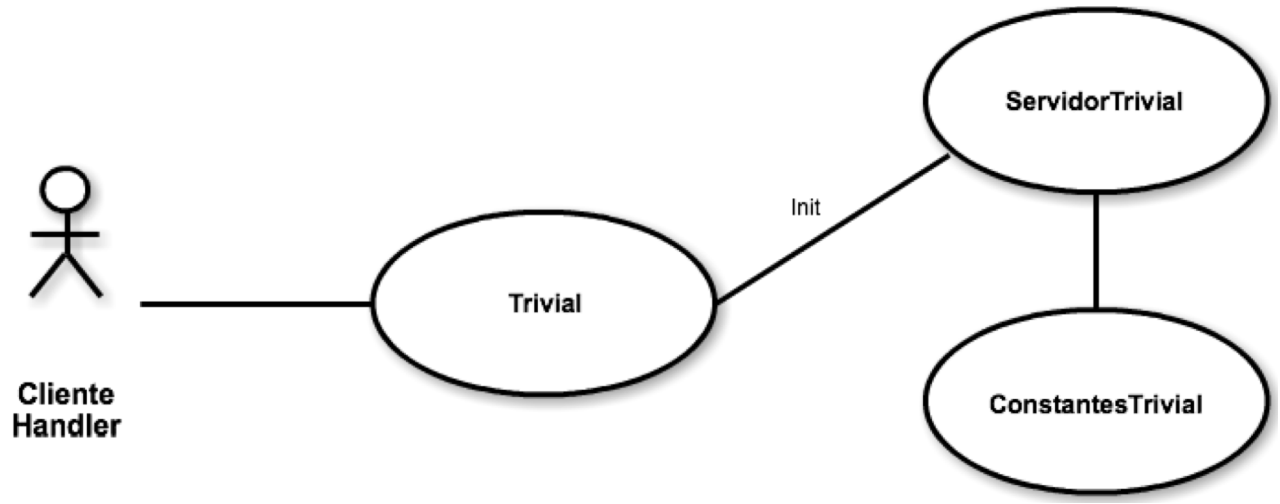
Caso de uso número 2: Nuevo cliente

Resumen de funcionalidad	Crea un nuevo cliente en el sistema.
Papel dentro del trabajo del usuario	Caso de uso frecuente .
Actores	Cliente.
Casos de uso relacionados	ClienteHandler.
Precondición	El cliente a registrar no existe.
Flujo normal	1- Tras entrar al juego y solicitarle identificarse, el cliente pulsa sobre la casilla de "Nuevo cliente" 2- El sistema le solicita nombre, Nick y password. 3- De manera transparente para el usuario, se registra en la base de datos, la hora, la fecha y la IP desde donde se conecta. 4- El cliente accede al juego.
Flujo alternativo	4- El sistema comprueba que todos los datos son correctos. En caso contrario, informamos al cliente y le pedimos que corrija el error. 4- El cliente cancela el alta .
Postcondición	
Aclaraciones	

Modulo Trivial

Este modulo permite la gestión de los nuevos clientes y su manera de entrar al juego. Previamente, el cliente debe estar registrado correctamente en el sistema.
El jugador esta siempre vinculado a su Nick y no puede participar dos veces con el mismo nombre.

La descripción gráfica de los casos de uso es la siguiente:



A continuación se detalla la descripción textual de los diferentes casos de uso de este modulo:
Caso de uso número 4: Trivial Registro

Resumen de funcionalidad	Solicita los datos del nuevo jugador.
Papel dentro del trabajo del usuario	Caso de uso muy frecuente ya que se usa para dar de alta nuevos clientes.
Actores	Cliente.
Casos de uso relacionados	
Precondición	Ninguna, es un nuevo cliente.
Flujo normal	1- Solicita Nick, password, nombre y mail 2- Acepta el registro.
Flujo alternativo	3- El Nick esta registrado anteriormente. 3- La longitud mínima de Nick o password no se cumple.
Postcondición	
Aclaraciones	

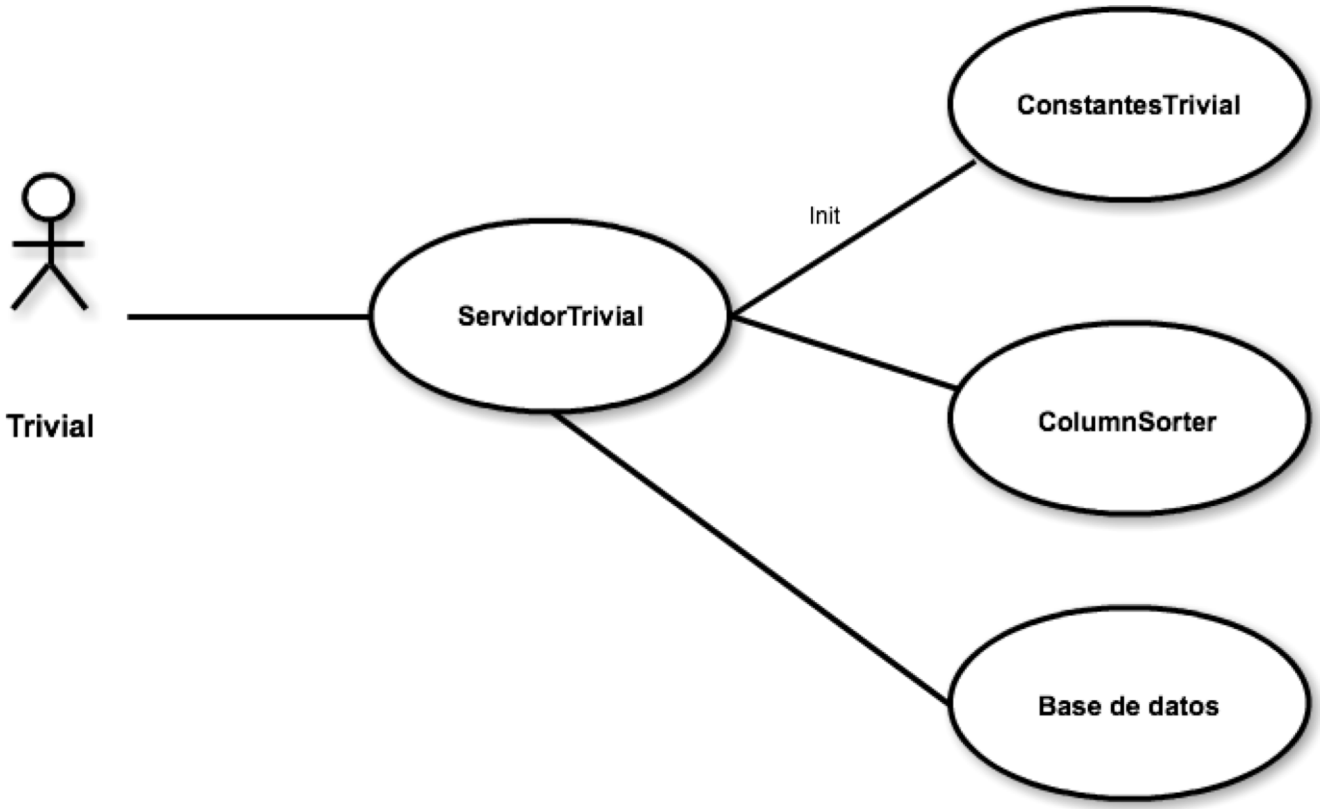
Caso de uso número 5: Nuevo Jugador

Resumen de funcionalidad	Entra un jugador en la sala de juego.
Papel dentro del trabajo del usuario	Caso de uso muy frecuente.
Actores	Cliente.
Casos de uso relacionados	Trivial Registro
Precondición	El cliente existe en el sistema.
Flujo normal	1- El cliente entra en la sala de juego. 2- El sistema inicializa el applet correspondiente al jugador.
Flujo alternativo	1- Si la clave es errónea, se informa al cliente. 3- Si el cliente cierra el navegador, el sistema lo elimina de la lista de usuarios en activo dentro del juego.
Postcondición	
Aclaraciones	

Módulo ServidorTrivial

Este modulo permite la gestión del juego, inicializando las tablas de preguntas y respuestas, acepta las conexiones de los clientes al juego, guarda los datos de los clientes en la base de datos y por ultimo, comprueba que las respuestas seleccionadas por los clientes son validas o no.

La descripción gráfica de los casos de uso es la siguiente:



A continuación se detalla la descripción textual de los diferentes casos de uso de este modulo:

Caso de uso número 6: Carga Preguntas

Resumen de funcionalidad	Se conecta a la base de datos y carga todas las preguntas.
Papel dentro del trabajo del usuario	Caso de uso poco frecuente, ya que solo se utiliza al arrancar el juego.
Actores	
Casos de uso relacionados	
Precondición	La base de datos debe estar ubicada en la carpeta indicada.
Flujo normal	1- Se inicializa el lanzador del sistema que llama al servidor. 2- El servidor se conecta con la base de datos. 3- El sistema informa de la correcta carga de preguntas.
Flujo alternativo	3 Ya hay una instancia ejecutándose. 3- El puerto esta ocupado.
Postcondición	
Aclaraciones	

Caso de uso número 7: Actualizar estadísticas

Resumen de funcionalidad	Modifica los datos de un cliente.
Papel dentro del trabajo del usuario	Caso de uso frecuente.
Actores	
Casos de uso relacionados	
Precondición	El cliente no existe en el sistema.
Flujo normal	1- El sistema conecta con la base de datos y chequea si el usuario existe. 2- El sistema prepara fecha y hora. 3- El sistema captura la IP de conexión. 4- El sistema guarda los datos en la base de datos
Flujo alternativo	5- Si la base de datos no es accesible, se lanza una excepción.
Postcondición	Se ha insertado un nuevo cliente.
Aclaraciones	

Caso de uso número 8: Preguntas

Resumen de funcionalidad	Carga las preguntas en el swing del juego.
Papel dentro del trabajo del usuario	Caso de uso muy frecuente.
Actores	
Casos de uso relacionados	Carga preguntas.
Precondición	Las preguntas están cargadas en el sistema.
Flujo normal	1- El sistema elige una pregunta al azar de la base de datos. 2- Se desordenan las respuestas ya que la correcta siempre la guardamos en la C. 3- Carga la pregunta en la tabla.
Flujo alternativo	1 No se encuentra el driver de la base de datos. 1- No se pudo realizar la conexión con la base de datos.
Postcondición	
Aclaraciones	

Caso de uso número 9: Respuestas

Resumen de funcionalidad	Comprueba las respuestas.
Papel dentro del trabajo del usuario	Caso de uso muy frecuente ya que se usa para comprobar la respuesta del cliente.
Actores	Cliente.
Casos de uso relacionados	Preguntas.
Precondición	El cliente a elegido una de las tres respuestas.
Flujo normal	1- El cliente ha marcado una de las respuestas y espera saber la solución. 2- El sistema comprueba con la clase preguntas, la ubicación de la respuesta correcta. 3- Si la respuesta es correcta, sumamos los puntos predefinidos en las constantes. 4- Si la respuesta es errónea, nos quita una de las vidas.
Flujo alternativo	5- Si el numero de vidas ha finalizado e informa al cliente de la finalización de su partida.
Postcondición	Suma puntos o resta vida.
Aclaraciones	

Material fuera del alcance del proyecto

En los diferentes casos de uso que se han detallado, se han definido todos los aspectos del juego pero existe una parte que quedan fuera del alcance del mismo.
La creación de un gestor de la base de datos desde el sistema para modificar los datos de los clientes, será el único punto que no se incluye en este proyecto, ya que la realización excede de la línea de tiempo que tenemos para terminar el proyecto y todos los cambios que se necesitan realizar se pueden hacer directamente desde la propia base de datos.

Otros requerimientos funcionales

En la clase "ConstantesServidor" se definen algunos puntos importantes del desarrollo del juego que pueden ser cambiados por el gestor del juego (y tras compilarlo de nuevo, debe lanzar nuevamente el juego), ya que informa al sistema de los siguientes puntos:

- Ruta de la Base de Datos.
- User y password de la Base de Datos.
- Puerto del servidor.
- Tiempo entre preguntas.
- Tandas de preguntas.
- Numero máximo de acertantes.
- Valor en puntos de la respuesta acertada.

Interface de usuario

Toda la aplicación se ejecuta desde un Applet que puede ser ejecutado desde cualquier navegador (Internet Explorer, Safari, Mozilla Firefox,...).

Un Applet es un programa escrito en Java y que forma parte de los componentes de una página de Internet. Los Applets han sido usados para proporcionar funcionalidad a páginas de Internet que no puede ser satisfecha usando únicamente HTML. La idea de los Applets es que sean lo suficientemente pequeños como para proporcionar una funcionalidad específica y claramente definida.

El uso de Applets en Java es muy conveniente ya que el código es independiente del sistema operativo en que se esté corriendo el navegador, sin importar si se trata de una PC con Windows, una Mac o una computadora con alguna de las variantes de Linux. El código de un Applet es ejecutado por el navegador mismo, usando lo que se llama la **máquina virtual Java (JVM**, por sus siglas en inglés: *Java Virtual Machine*).

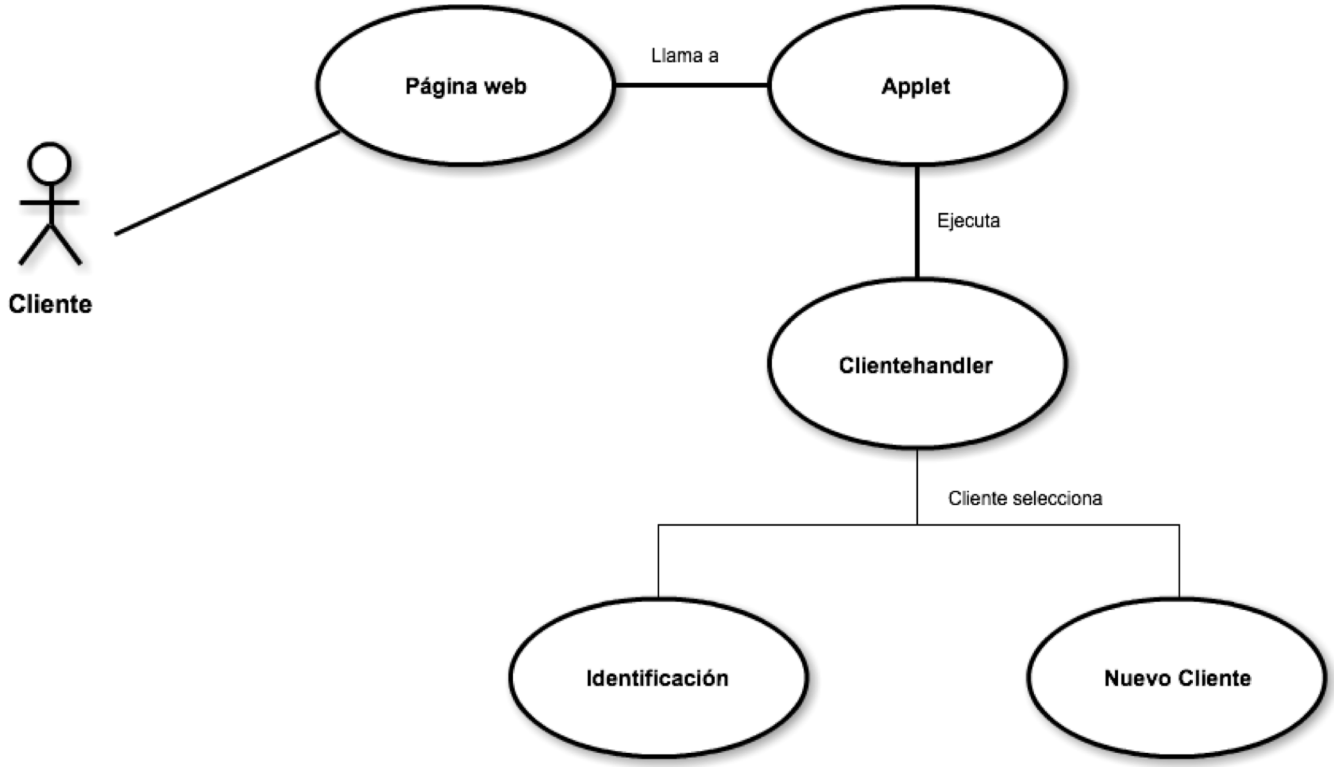
Algunos ejemplos de funcionalidad de páginas de Internet que usan Applets son:

- Funciones especializadas, como por ejemplo, Applets para calcular el valor del ángulo inscrito en una circunferencia y circuncentro de un triángulo.
- Mostrar secuencias de imágenes y agregar efectos visuales.
- Mostrar imágenes con sonidos y agregar efectos sonoros.
- Permitir la presentación de gráficos interactivos, reaccionando a acciones que se toman con el mouse sobre el gráfico.
- Animaciones de textos y efectos especiales sobre los mismos.
- Crear diagramas y gráficas, como por ejemplo la clásica gráfica de rebanadas de pastel.
- Juegos sencillos.

Diagrama de flujo entre pantallas.

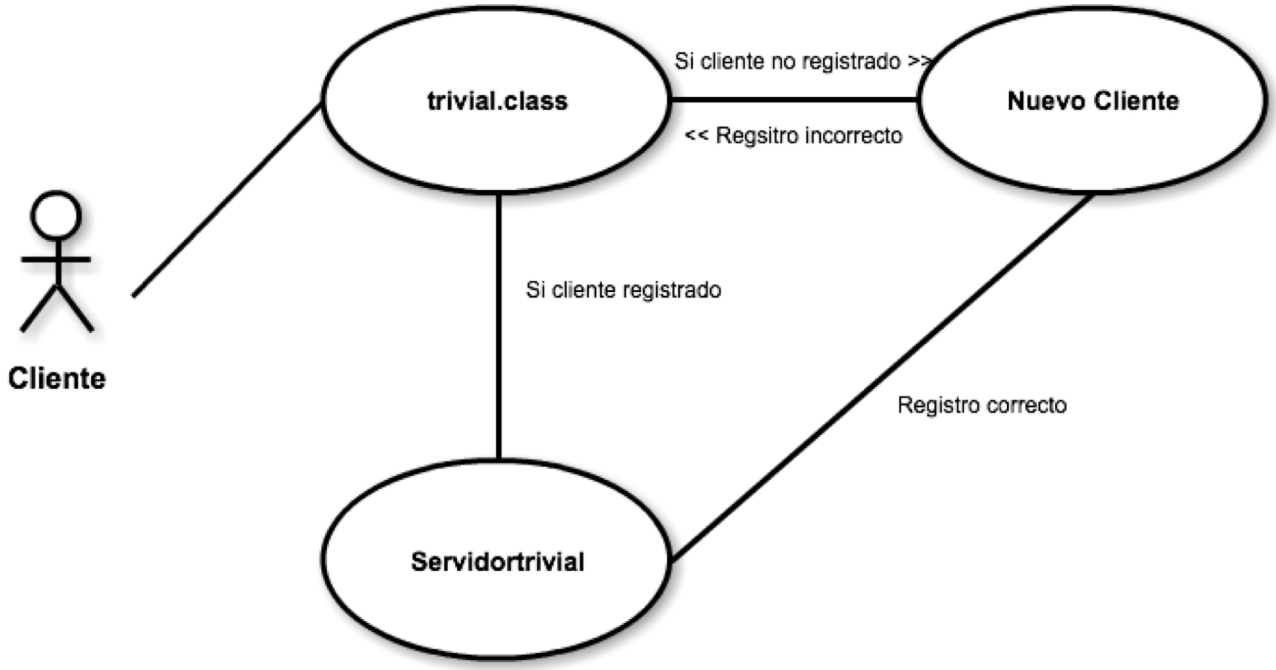
Módulo ClienteHandler.

El nuevo jugador entra en la página web y le aparece una pantalla que le solicita identificarse, a través de un nombre de usuario y una contraseña. En el caso que no este registrado deberá pulsar sobre la opción "Nuevo usuario" donde le solicitara su nombre, correo electrónico, Nick de usuario y una contraseña.
Tras haber rellenado el formulario, el cliente accede a la pagina principal del juego. Mas adelante, en otro modulo, se comprobara si el Nick del jugador esta ocupado o si cumple las normas en cuanto a tamaños y tipo de caracteres utilizados.



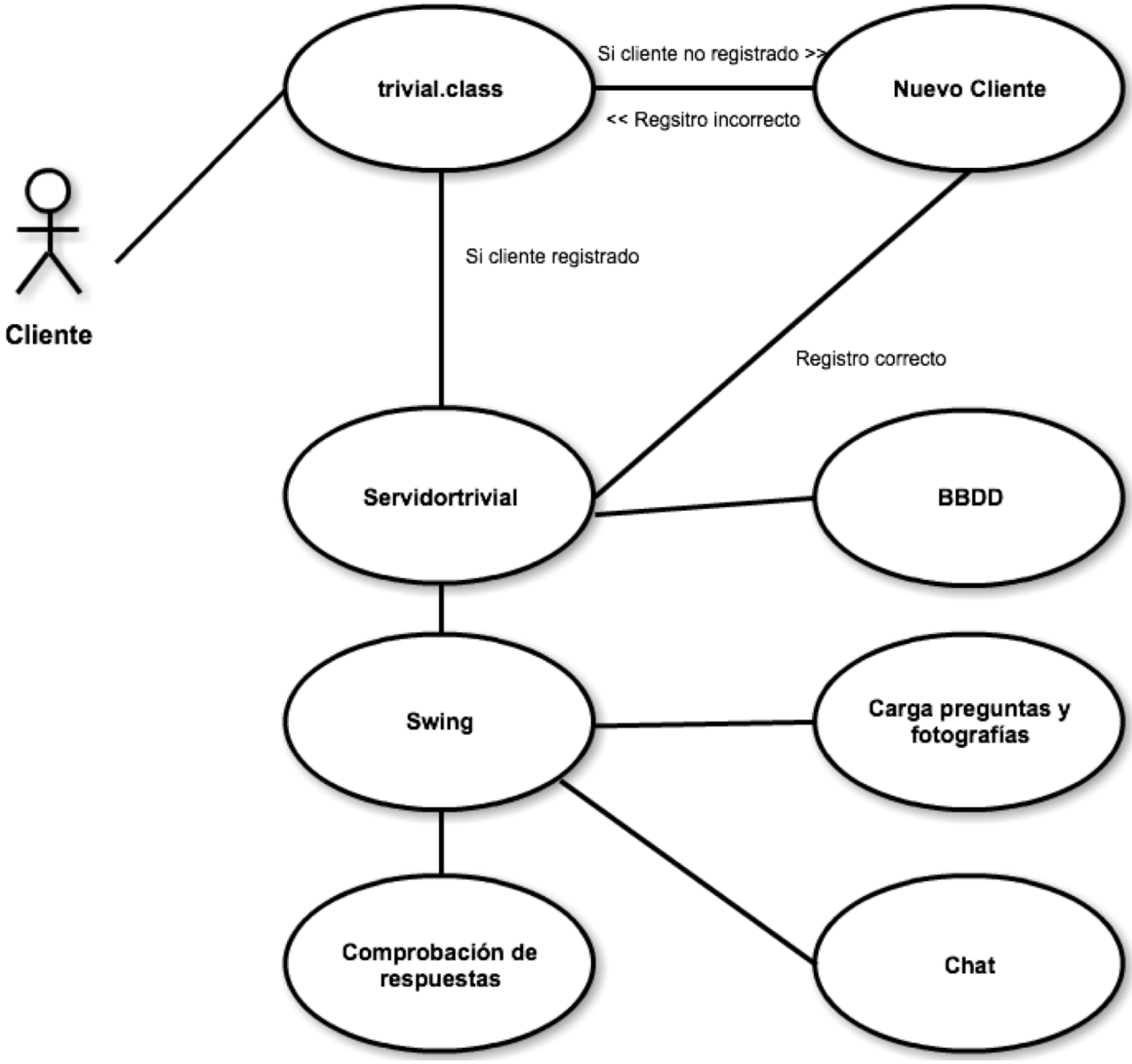
Módulo Trivial.

El módulo Trivial gestiona la correcta identificación del cliente, tanto si es nuevo como si ya esta identificado. Comprueba que el registro es correcto y se da de alta en la base de datos y por otro lado llama al main del juego para que entre en la sala que esta en juego actualmente.



Módulo ServidorTrivial.

Esta es la parte mas importante del programa ya que se encarga de conectarse a la base de datos y cargar todas las preguntas y fotografías, carga las preguntas y las formatea adecuadamente en la pantalla de juego, realiza consultas de clientes contra la base de datos, comprueba las respuestas y gestiona el chat entre jugadores.



Diseño de clases

A continuación se describen las clases que intervienen en la aplicación.

ClienteHandler
+clienteHandler +MyTableModel +MyTableModel2

Explicación de los métodos:

Método	**Descripción**
Trivial	Clase principal del Applet
pedirFotografia	Método que captura fotografías del servidor web
notificarSalida	Envia una trama de aviso indicando que hemos salido del juego
creaCliente	Crea una clase Clientehandler que contiene la lógica de la aplicacion
stop	Se ejecuta cuando el usuario cierra el navegador y envía trama de desconexión al servidor
registrarUsuario	Envia trama para poder registrar a un nuevo usuario
getURL	Devuelve la URL absoluta para poder realizar la carga en un objeto de imagen

clienteHandler
-estiloOriginal
-estiloVerde
-texto
-creaEditor
-creaEstilo
+tratarEnvioChat
+enviarResultados
+notificarEntrada
+notificarSalida
+desactivarBotones
+tratarInformacionPuntuaciones
+tratarInfoPreguntasRespuestas
+mostrarFotografia
+numAzar
+tratarInformacionUsuarios
+tratarPuntuaciones
+tratarRecords
+pasarAVector
+analizarTrama
+contarColumnas
+elementosArray
+nombrePedido
+tratarInformacionChat

Explicación de los métodos:

Método	Descripción
estiloVerde	Estos 2 métodos permiten cambiar
estiloOriginal	el color de la letra
texto	Crea un nuevo JTextPane
creaEditor	Crea un Jtextpane de estilos
creaEstilo	Crea un nuevo StyleContext
tratarEnvioChat	Enviamos al servidor una trama con el mensaje de chat
enviarResultados	Procesa la respuesta y envía el resultado al servidor
notificarEntrada	Envía trama de aviso de entrada
notificarSalida	Envía trama de aviso de salida
desactivarBotones	Desactiva los botones de respuesta cuando ha terminado el tiempo
tratarInformacionPuntuaciones	Trata la puntuación dependiendo si han acertado o no
tratarInformacionpreguntasRespuestas	Trama con información sobre las preguntas
mostrarFotografia	Obtiene una nueva foto y la muestra
numAzar	Devuelve un numero aleatorio entre 0 y -1
tratarInformacionUsuarios	Procesa la información de los usuarios entrantes
tratarPuntuaciones	Procesa la subtrama de puntuaciones y ordena de mayor a menor
pasarAVector	Con la información ordenada, la pasamos a los dos vectores de un jtable y la muestra en pantalla
tratarRecords	Trata la subtrama de records y la inserta en la jtable
analizarTrama	Analiza las tramas entrantes identificando el tipo
contarColumnas	Devuelve el numero de elementos del array de nombres de columnas
elementosArray	Devuelve el numero de elementos del array de datos
nombrePedido	Devuelve el elemento indicado

ColumnSorter
+compare

ServidorTrivial
+ServidorTrivial +ActualizarEstadisticas +CargarPreguntas +numAzar

Explicación de los métodos:

Método	Descripción
actualizarEstadisticas	Función que guarda la información de los usuarios conectados en la Base de Datos
cargarPreguntas	Carga las preguntas de la base de datos, desordena las preguntas y recoge los nombres de los máximos acertantes
numAzar	Devuelve un numero aleatorio entre 0 y “numero”

Trivial
+NotificarSalidaSinNick +Stop +registrarUsuario +Autentificacion +URL GetURL

Explicación de los métodos:

Método	Descripción
NotificarSalidaSinNick	Función que envía una trama informando que hemos salido de la aplicacion
stop	Se ejecuta cuando el cliente cierra el navegador
registrarUsuario	Envia trama de registro de usuario
Autentificacion	Envia temática, user y pass al servidor
URL getURL	Recibe el nombre de la fotografía que debe cargar y devuelve la URL absoluta

ServidorHandler
+ServidorHandler +Respuestas

ServidorHandler
+iniciaTablaPuntuaciones +iniciaTablaNombresMaximos +iniciaTablaPuntosMaximos +trataTablaUsuarios +trataSalidaUsuarios +trataEntradaUsuarios +trataRespuestas +actualizaRecords +realizaRegistro +caracterValido +comprobarAutentificacion +cargaRecords +numeroAzar +enviarListadoUsuarios +analizarTrama

Preguntas
+obtenerCadenaRecords -broadcast +enviarDatos

Respuestas
+enviarRespuestas

Explicación de los métodos:

Método	Descripción
iniciaTablaPuntuaciones	Crea un array de hashmaps para guardar los pares jugador-puntuación
iniciaTablaNombresMaximos	Crea un array de arrays para guardar los pares jugador-puntuación
iniciaTablaPuntosMaximos	Crea un array de arrays de puntuación-jugador
trataTablaUsuarios	Crea un array de arrays para guardar los puntos de los máximos acertantes
trataSalidaUsuarios	Trata la trama que recibe el servidor cuando sale un nuevo jugador
trataEntradaUsuarios	Trata la trama que recibe el servidor cuando entra un nuevo jugador
trataRespuestas	Función que trata la trama de respuestas
actualizaRecords	Cuando un usuario acierta una pregunta, comprueba si ha hecho record y actualiza la BBDD
realizaRegistro	Recoge la temática, user, pass, nombre, mail y comprueba que existe en la BBDD y lo da de alta en la misma
caracterValido	Comprueba si el usuario ha introducido algún carácter invalido, evitando posibles vulnerabilidades
comprobarAutentificacion	Función que recoge user y pass y mira si existe en la BBDD
cargaRecords	Carga los records de la BBDD y busca los records de la tabla usuarios
numeroAzar	Devuelve un numero entre 0 y numero
enviarListadoUsuarios	Envía a los clientes información actualizada de los usuarios que están conectados
analizarTrama	Analiza las tramas que envía el cliente y devuelve un entero que indica el tipo de trama

Diseño de la base de datos

A continuación se muestra el diseño de la base de datos.
Se muestra el diseño lógico relacional, mas cercano a la implementación de un sistema gestor de base de datos relacional.

Diseño lógico de la base de datos.

ENTIDADES

ESTADISTICA
(
num , ip(50), dia, mes, año, hora
)

PREGUNTAS
(
id, pregunta(255), resp1(100), resp2(255), resp3(100), categoría(50)
)

USUARIOS
(
nick(30), password(30), nombre(60), email(60), maxuoc(50)
)

NOTAS

- Las bases de datos son independientes entre sí, por lo que no tienen ninguna relación entre ellas ya que se busca la sencillez del diseño, ofreciendo una tabla donde el programa recoge las preguntas y respuestas, otra tabla donde están los usuarios con su máximo record y por último la tabla de estadística donde se ubican los datos de las conexiones para tener un buen control de los usuarios.

TABLAS

A continuación se muestra el diseño de las tablas de la base de datos y una breve descripción.

Tabla Estadísticas

Permite almacenar los datos de la conexión de un usuario.

Campo	Tipo	CV	NULL	Descripción
Num	Autonumérico	Mk	No	Almacena el numero de conexión.
ip	Texto		No	Guarda la dirección IP del cliente.
Dia	Integer		Si	Día de la conexión.
Mes	Integer		Si	Mes de la conexión.
Año	Integer		Si	Año de la conexión.
Hora	Fecha/Hora		Si	Hora de la conexión.

Tabla Preguntas

Permite almacenar las preguntas y respuestas del juego.

Campo	Tipo	CV	NULL	Descripción
id	Autonumérico	Mk	No	Identificador de pregunta (evita repeticiones)
Pregunta	Char		Si	Almacena la pregunta del juego.
Resp1	Char		Si	Respuesta 1 a la pregunta.
Resp2	Char		Si	Respuesta 2 a la pregunta.
Resp3	Char		Si	Respuesta 3 a la pregunta. (Correcta).
categoría	Char		Si	Categoría de la pregunta.

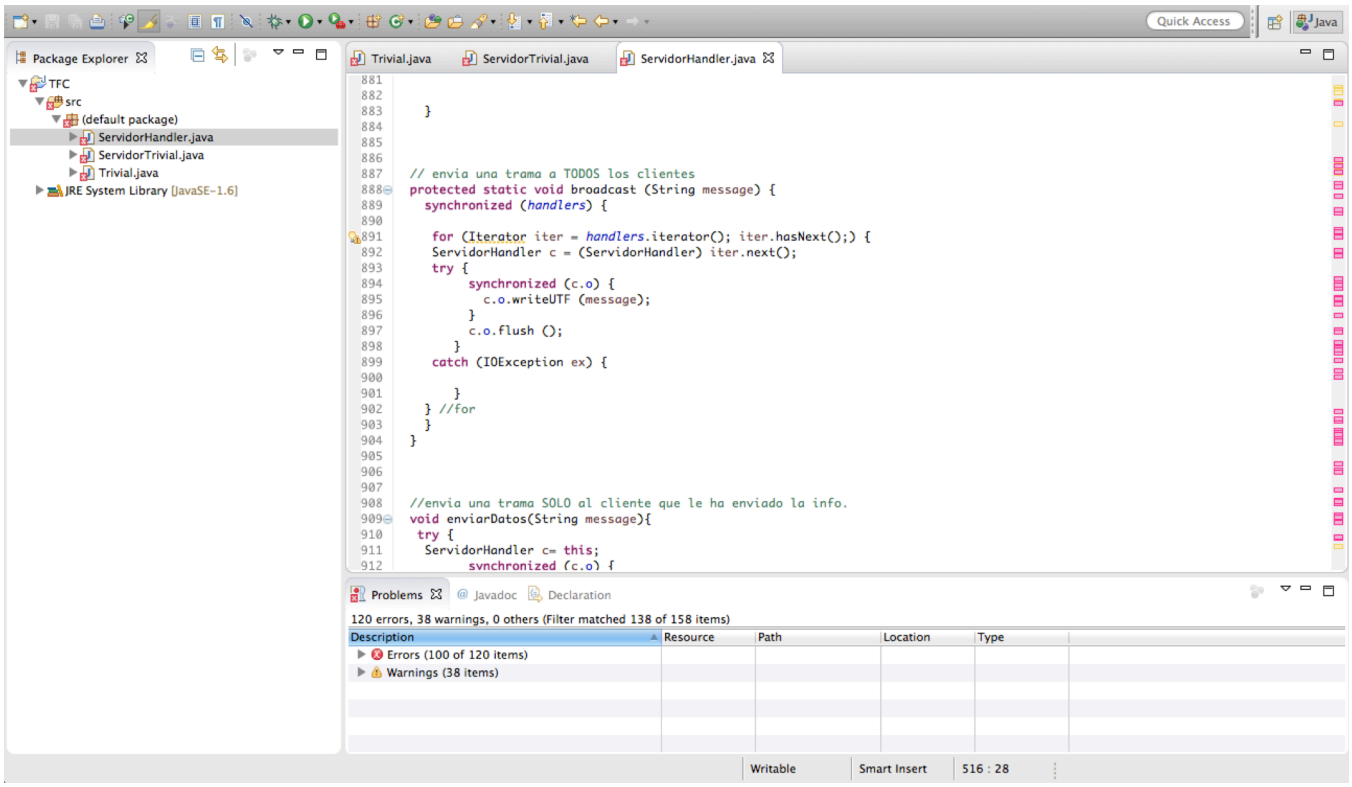
Tabla Usuarios

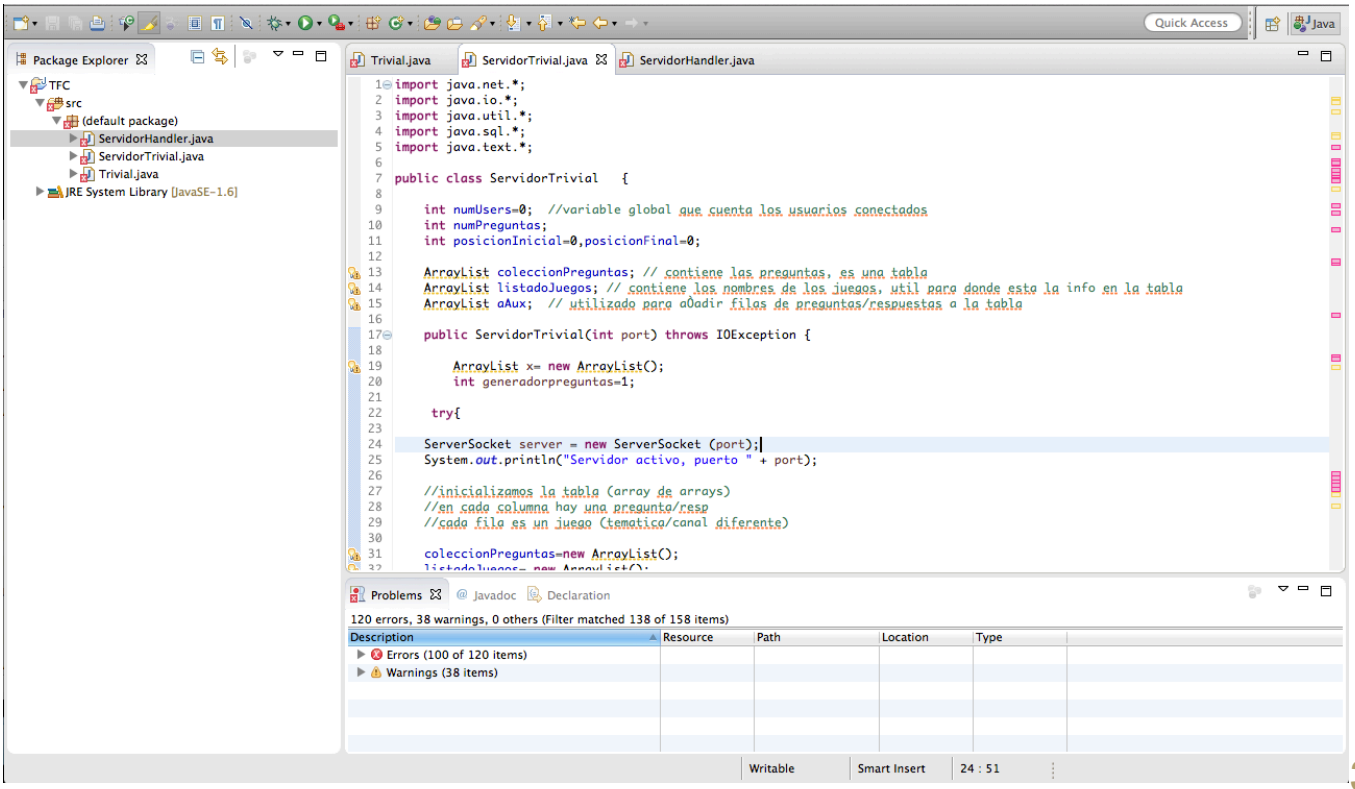
Permite almacenar los usuarios registrados en el sistema.

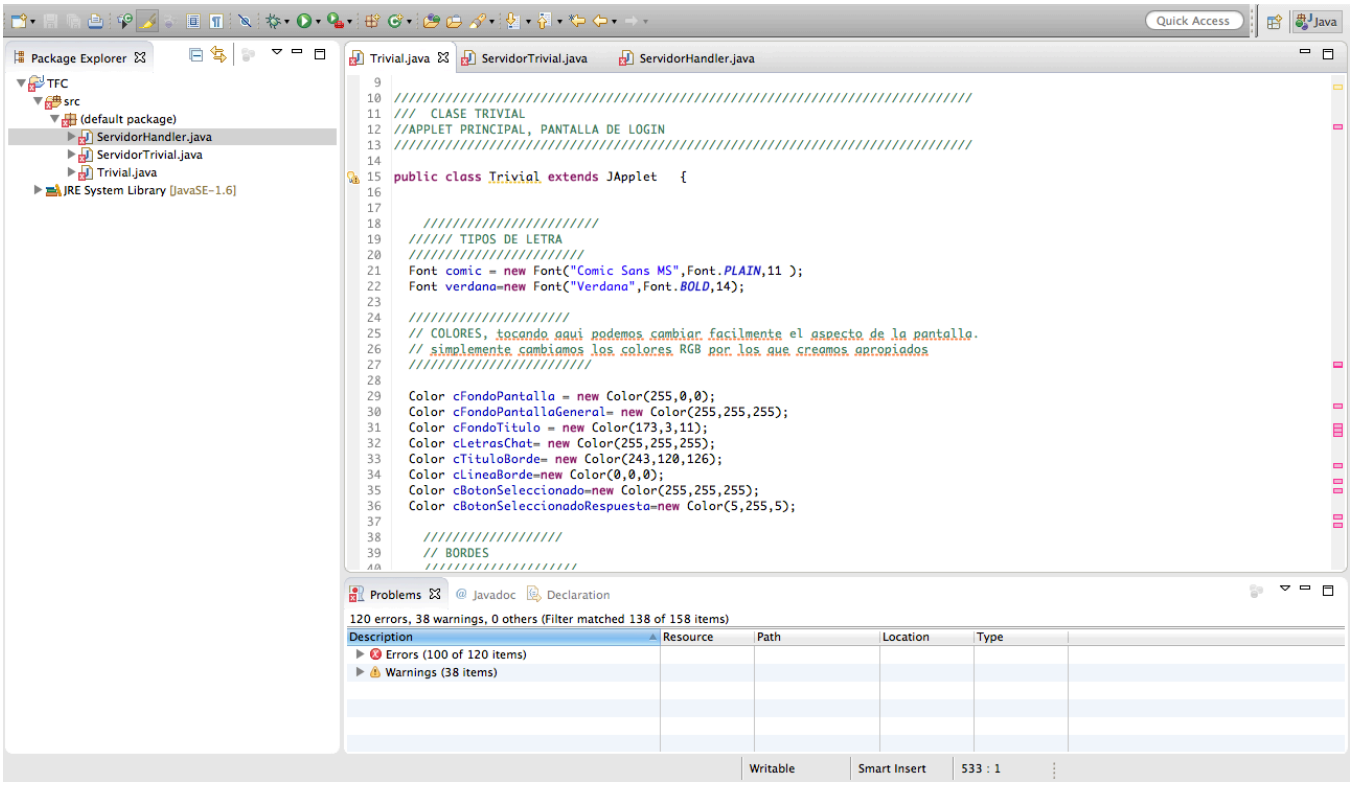
Campo	Tipo	CV	NULL	Descripción
Nick	Char	Mk	No	Nick del jugador.
Password	Char		No	Password del jugador.
Nombre	Char		No	Nombre del jugador.
Email	Char		No	Email del jugador
maxuoc	Char		Si	Máxima puntuación en sus partidas.

Anexo

Captura de las tecnologías utilizadas:

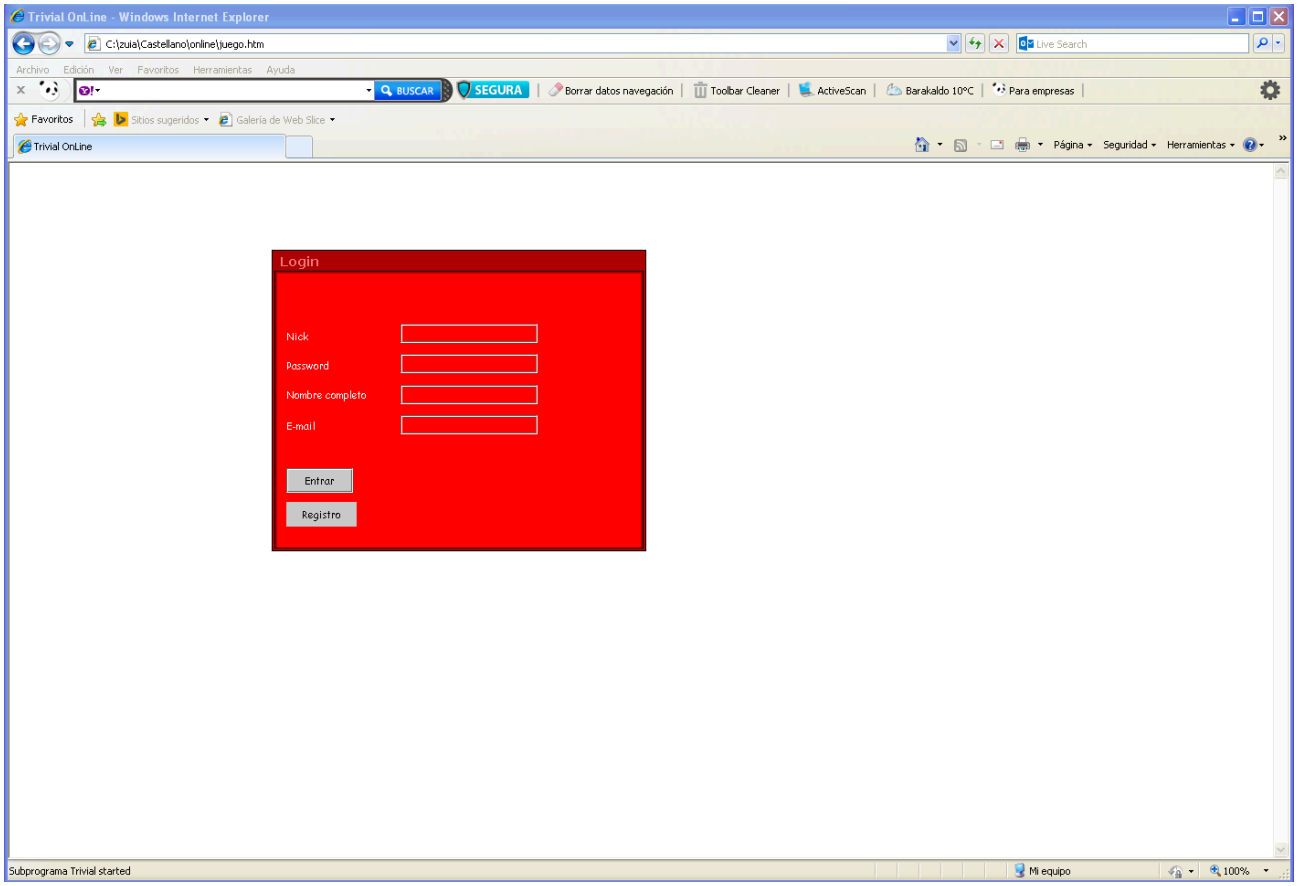




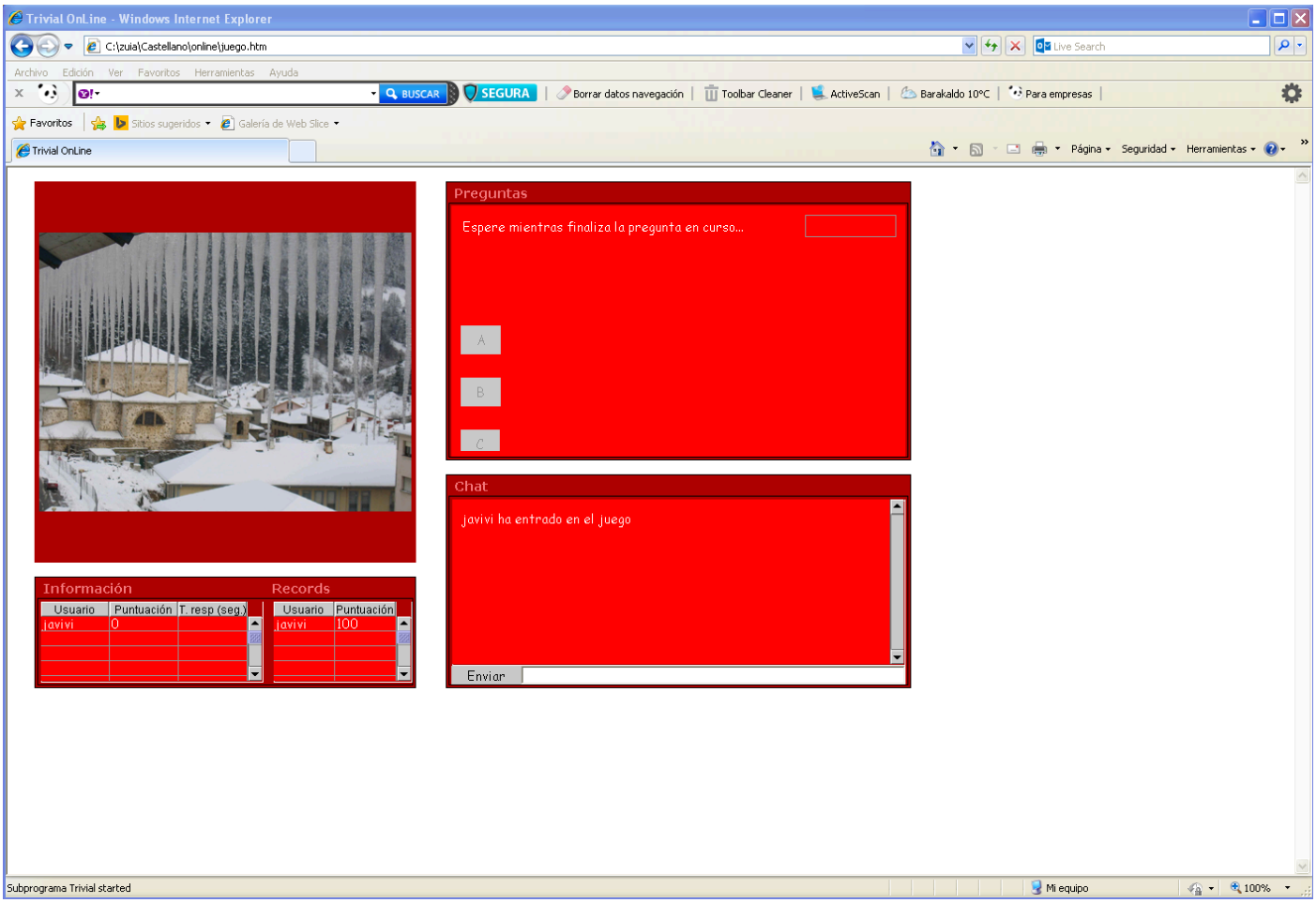

Package Explorer
TFC
src
(default package)
ServidorHandler.java
ServidorTrivial.java
Trivial.java
JRE System Library [JavaSE-1.6]
Trivial.java
ServidorTrivial.java
ServidorHandler.java
Quick Access
Java
public class Trivial extends JApplet {
Problems
Javadoc
Declaration
120 errors, 38 warnings, 0 others (Filter matched 138 of 158 items)
Description
Resource
Path
Location
Type
Errors (100 of 120 items)
Warnings (38 items)
Writable
Smart Insert
533 : 1

Resultado final del proyecto:

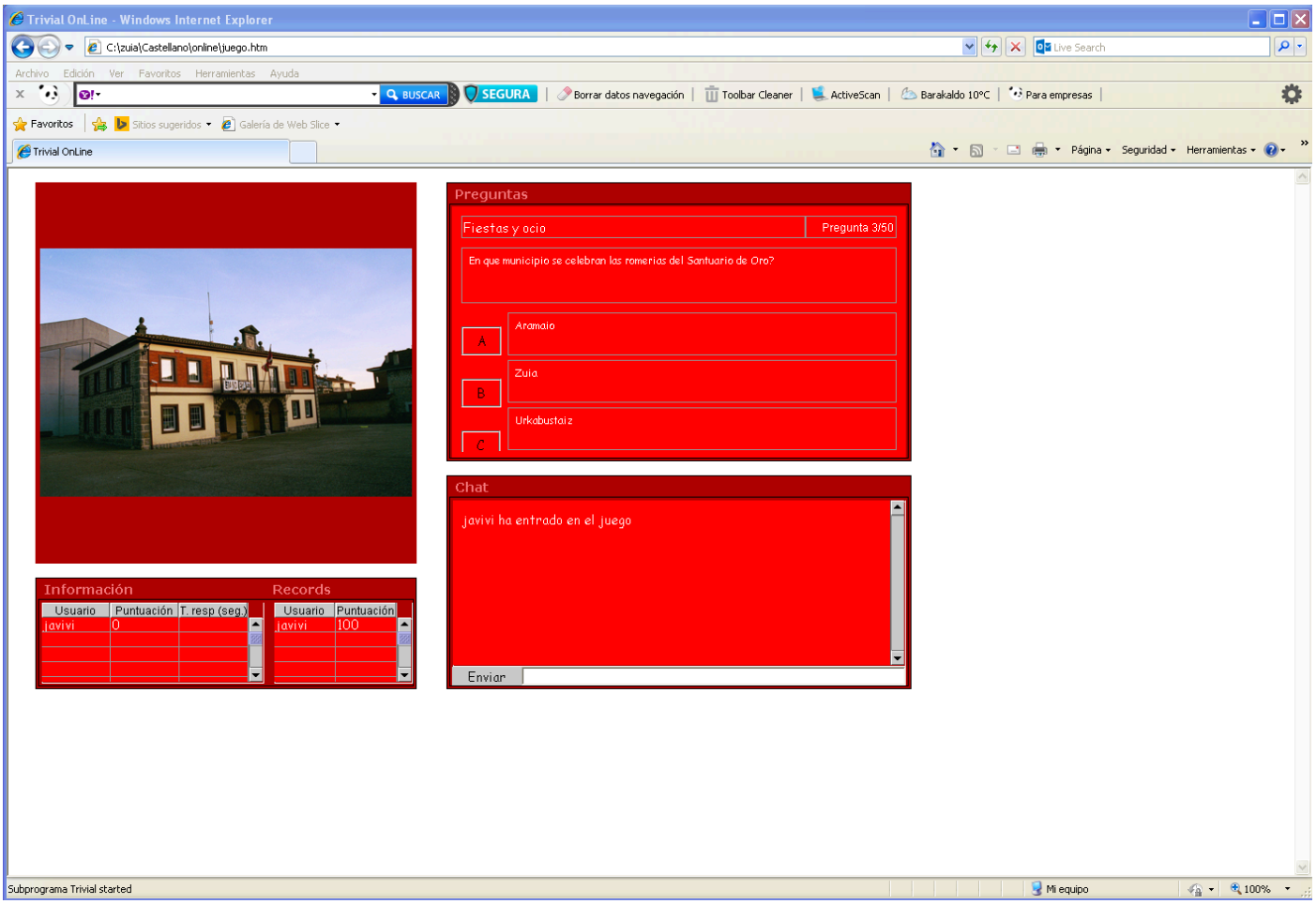
Acceso por primera vez a la aplicación, hay que registrarse rellenando todos los campos, la próxima vez que se accede tan solo hay que rellenar Nick y password.



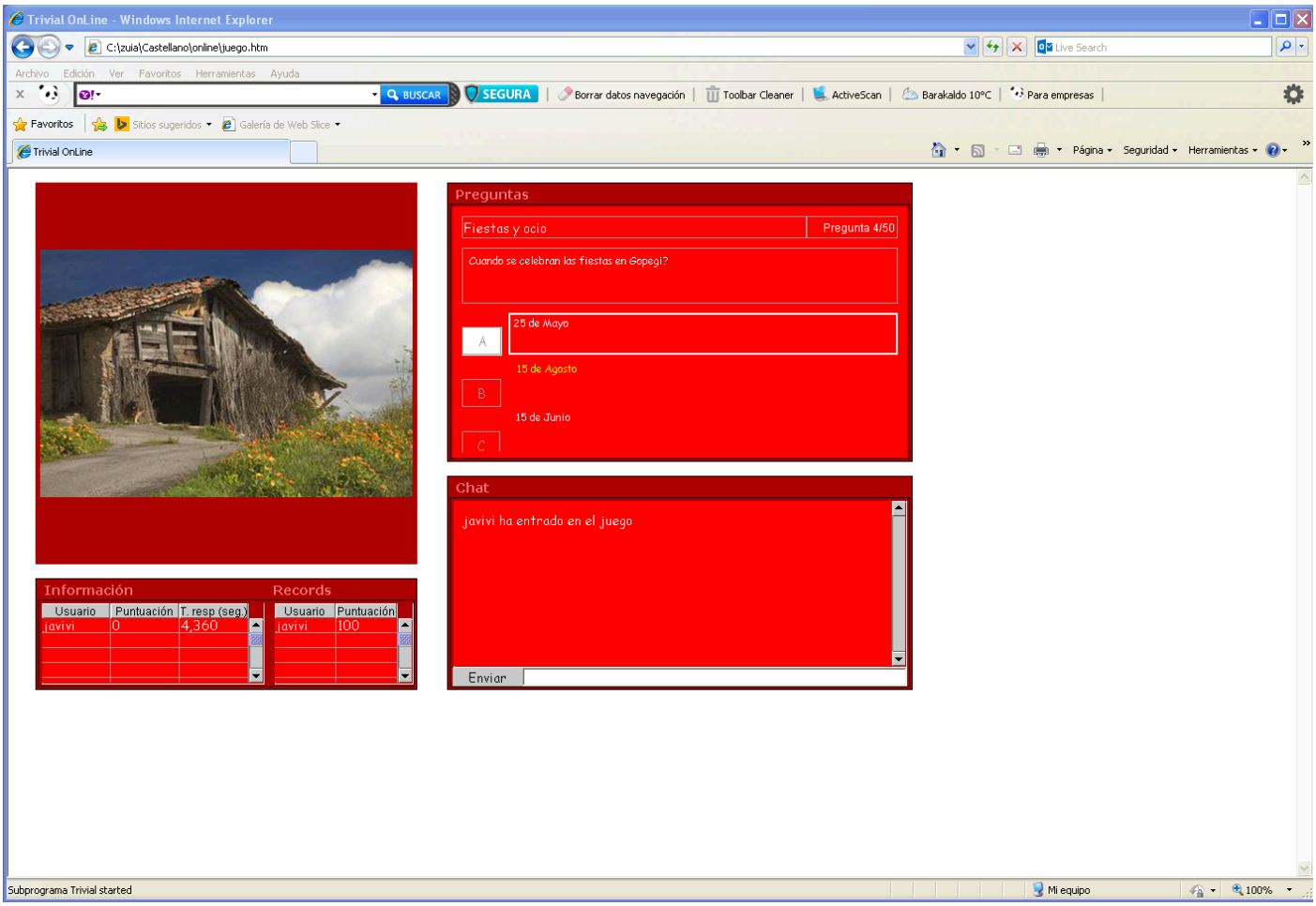
Entrada al juego, hay que esperar que termine la pregunta en curso para comenzar a jugar.



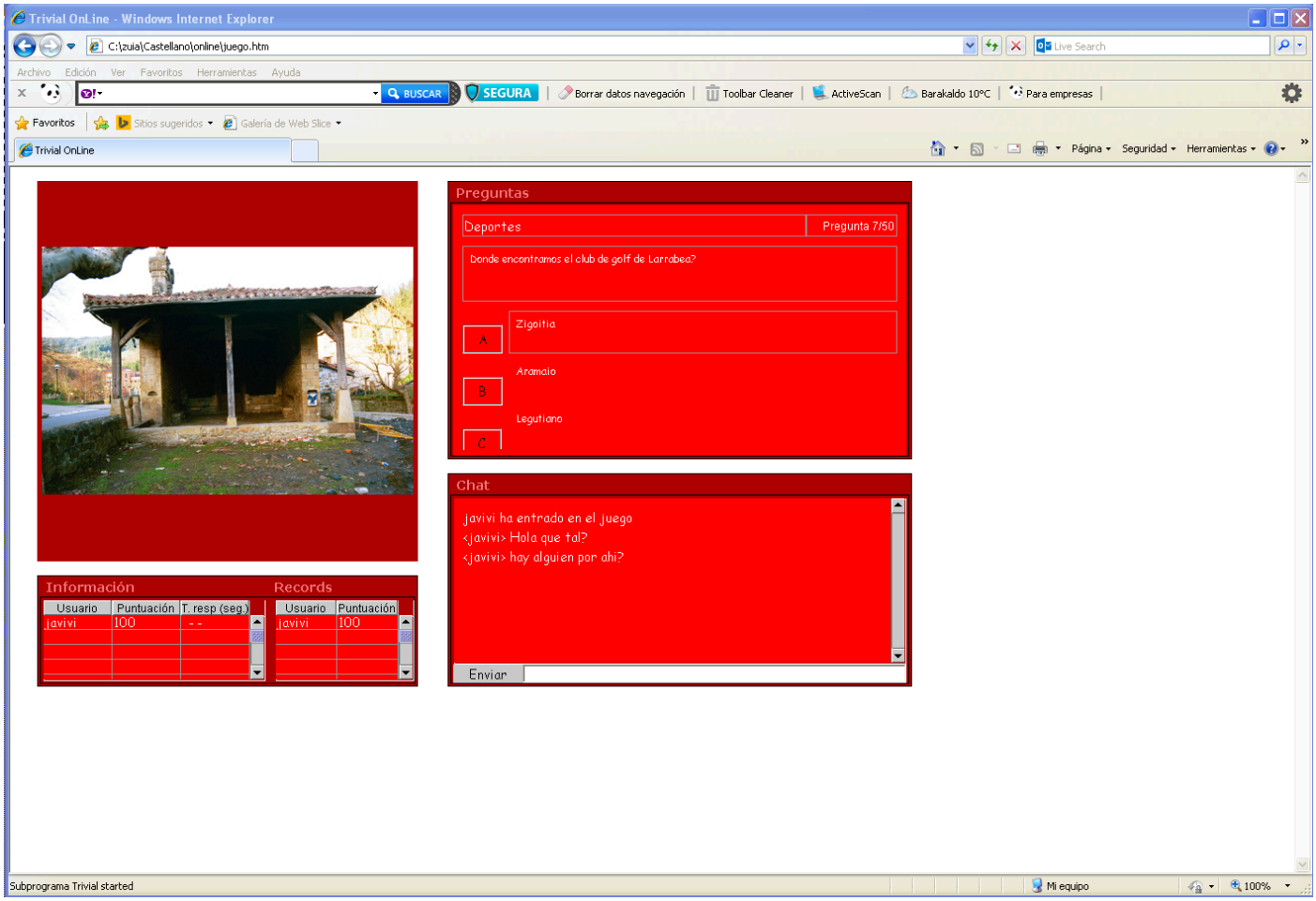
La aplicación muestra la siguiente pregunta, tres posibles respuestas, así como, una fotografía relacionada. En la parte inferior puede observarse la tabla de puntuaciones y la zona de chat.



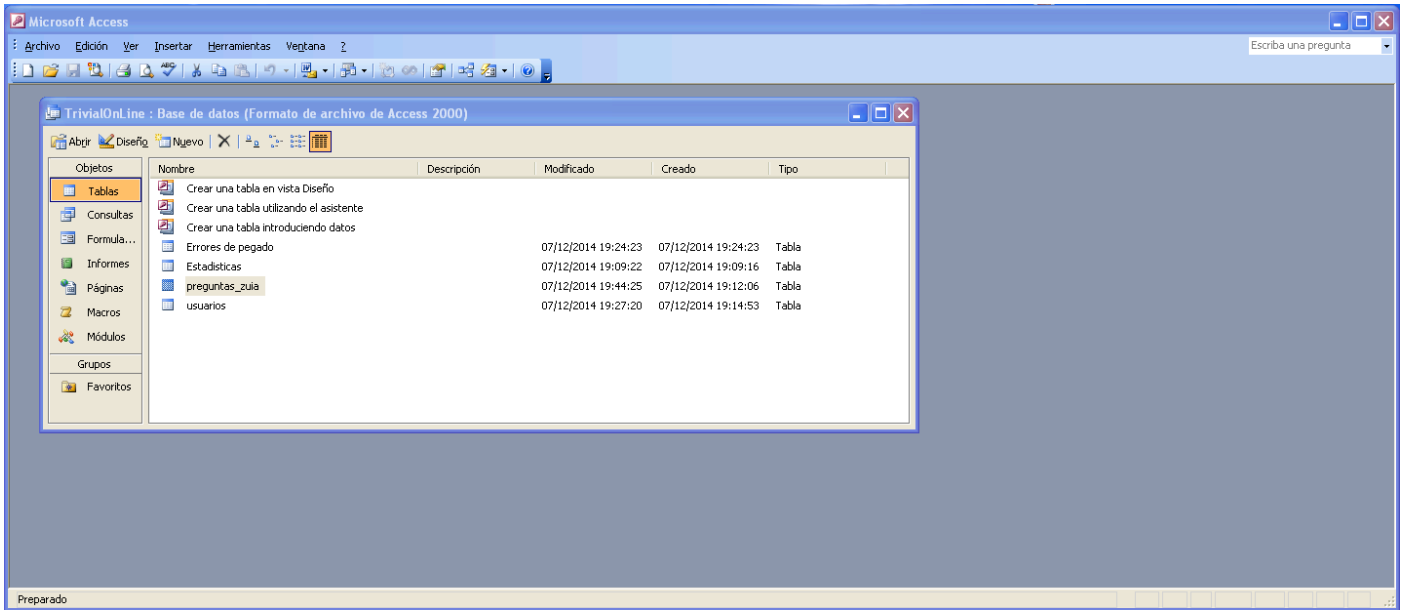
En esta ocasión, se ha seleccionado la respuesta A, pero podemos observar como el juego nos marca en amarillo que la respuesta correcta es la B.

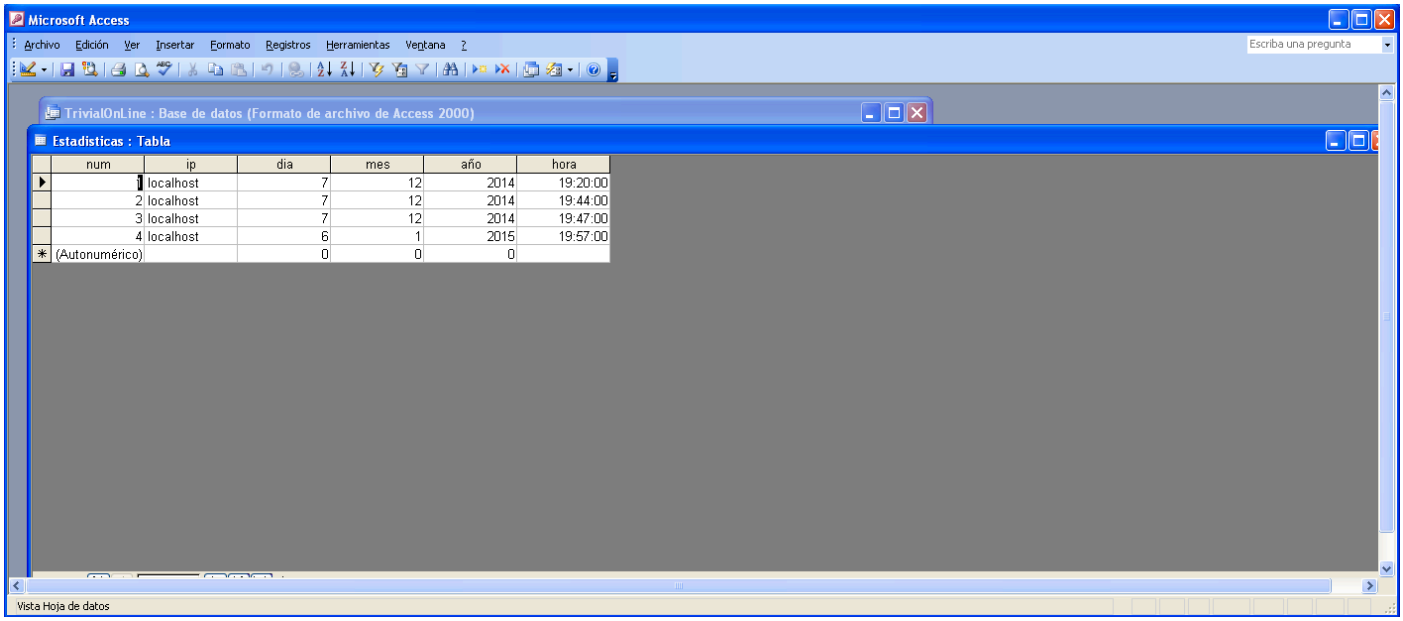


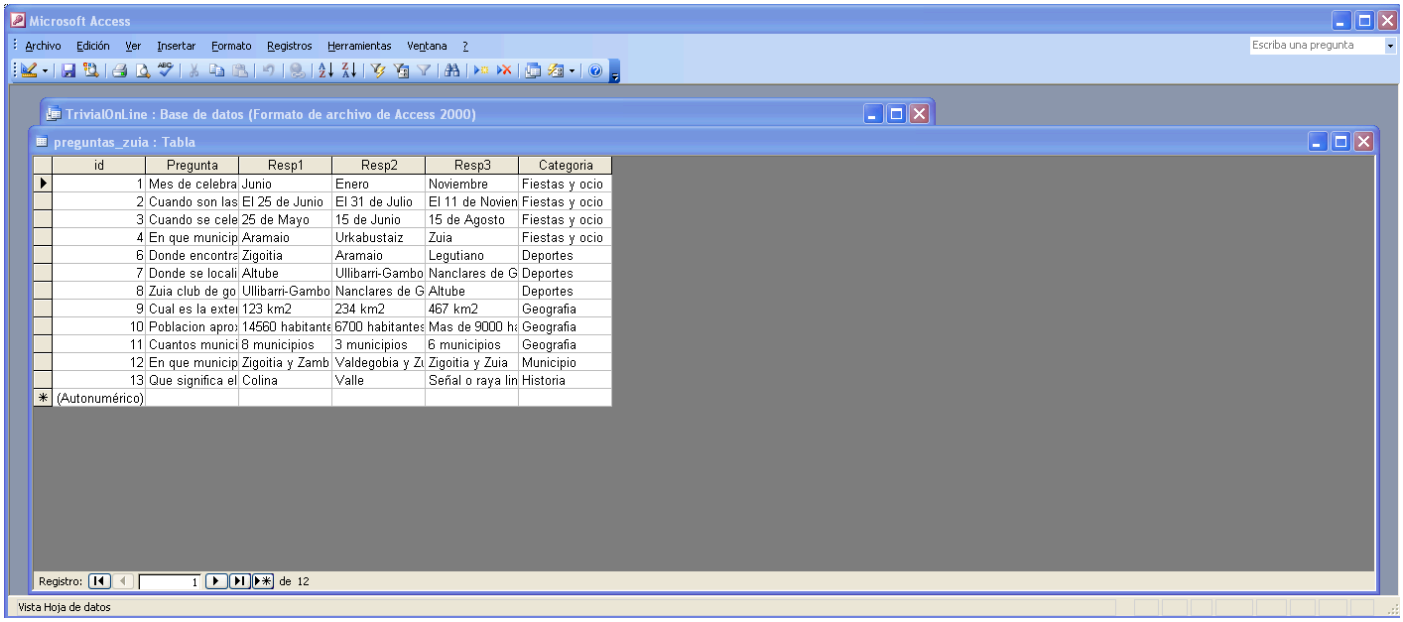
En esta captura se puede ver, en la parte inferior derecha, como el jugador esta chateando con el resto de participantes (si existiese alguno).



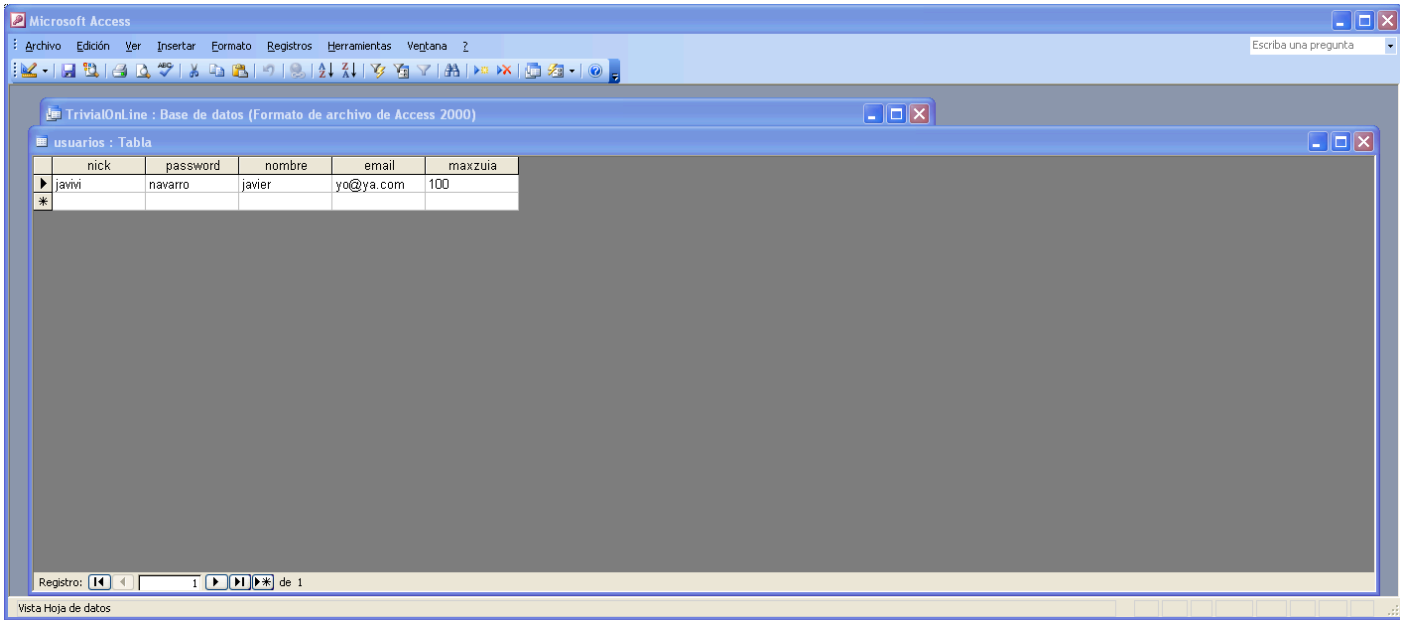
En las siguientes capturas podemos visualizar la Base de Datos con las diferentes entidades y su contenido. Desde aquí, se puede gestionar las preguntas y respuestas, eliminar usuarios, consultar fechas, etc ...





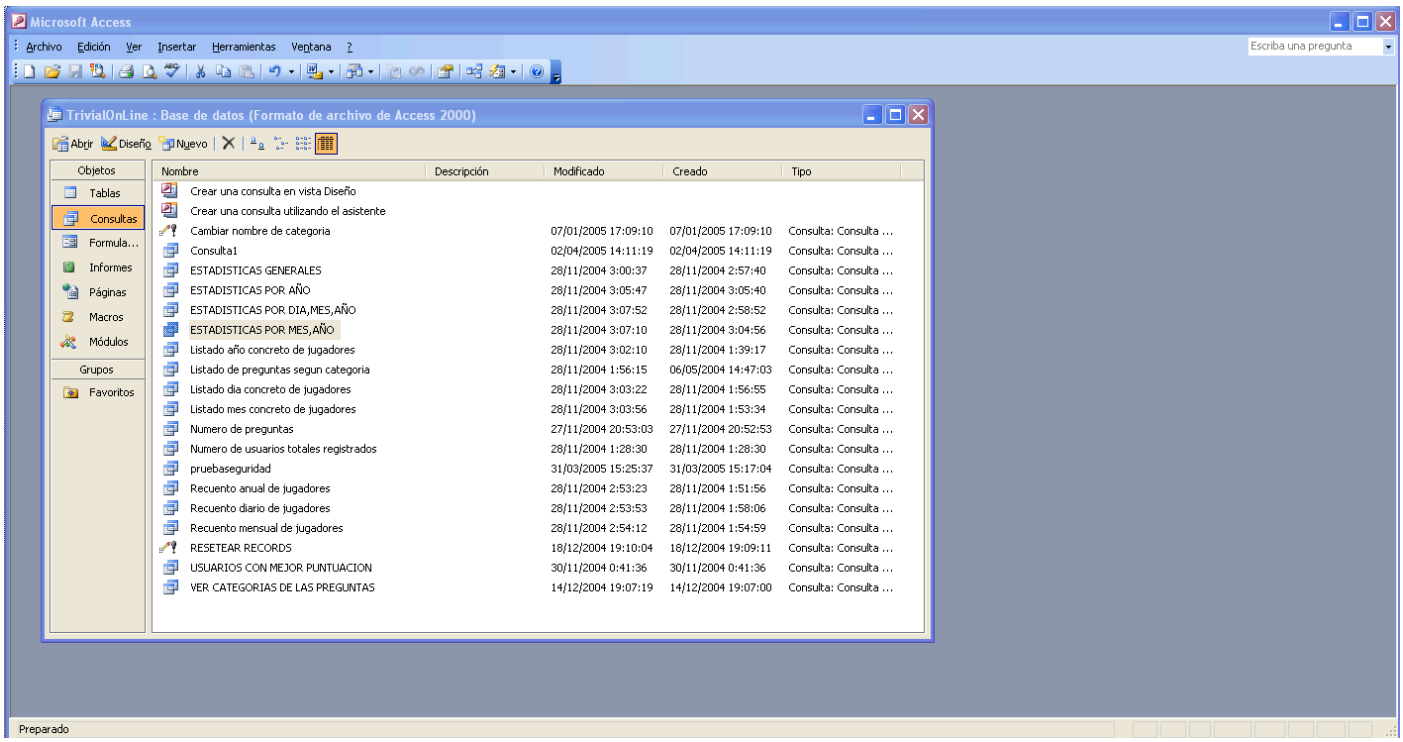


id	Pregunta	Resp1	Resp2	Resp3	Categoria
1	Mes de celebra	Junio	Enero	Noviembre	Fiestas y ocio
2	Cuando son las	El 25 de Junio	El 31 de Julio	El 11 de Novien	Fiestas y ocio
3	Cuando se cele	25 de Mayo	15 de Junio	15 de Agosto	Fiestas y ocio
4	En que municip	Aramaio	Urkabustaiz	Zuia	Fiestas y ocio
6	Donde encontra	Zigoitia	Aramaio	Legutiano	Deportes
7	Donde se locali	Altube	Ullibarri-Gambo	Nanclares de G	Deportes
8	Zuia club de go	Ullibarri-Gambo	Nanclares de G	Altube	Deportes
9	Cual es la exte	123 km2	234 km2	467 km2	Geografia
10	Poblacion apro	14560 habitante	6700 habitantes	Mas de 9000 h	Geografia
11	Cuantos munici	8 municipios	3 municipios	6 municipios	Geografia
12	En que municip	Zigoitia y Zamb	Valdegobia y Z	Zigoitia y Zuia	Municipio
13	Que significa el	Colina	Valle	Señal o raya lin	Historia
(Autonumérico)					



nick	password	nombre	email	maxzuia
javivi	navarro	javier	yo@ya.com	100

En el apartado consultas, se han dejado programadas una serie de comandos SQL para poder acceder a estadísticas, listados, recuentos, reset y un largo etc.



Requisitos

Para ejecutar este juego, se debe utilizar un sistema operativo Windows XP con Office 2003. Las preguntas están almacenadas en una base de datos Access donde se conecta el Applet, realiza la carga de las preguntas, almacena los usuarios y los records.

El juego debe cargarse (la carpeta completa Zuia) en la raíz de C. El juego esta compilado contando con que la base de datos este en esa ubicación, de lo contrario, no funcionara.

Este desarrollo está basado en la Cuadrilla de Zuia, municipios Alaveses, lugar donde paso las vacaciones con mi familia de donde son naturales. Sirva este juego a modo de homenaje a ellos.

Esta compilación se ha depurado de tal manera que con unos pequeños cambios se puede adaptar a los fines que queramos, como educación, cultura, formación, ... En la carpeta "Juego adaptado" hay una muestra con un cambio del juego en lo referente a colores, contenidos, etc basado en la UOC. De la misma manera, si se observa la estructura de las carpetas del juego, se puede ver que una de ellas se llama castellano, dando la posibilidad de separar también por idiomas el juego con unos pequeños cambios en el código.

Así mismo, se puede usar un servidor web tipo XAMPP (se incluye en la carpeta "Herramientas") que lance la página index.html y abriendo los puertos necesarios (4322) en el router donde este instalado, podemos tener un juego online donde pueden participar jugadores de todo el mundo y comunicarse con el Chat que lleva incluido el juego.

Funcionamiento

La mecánica del juego es la siguiente:

1- Una vez ubicada la carpeta en la raíz del disco C, ejecutamos el archivo "lanzarservidor" ubicado dentro de zuia/castellano/online/

2- A continuación, ejecutamos index.html ubicado en el mismo lugar.

3- La primera página nos pedirá identificarnos pero si no estamos registrados en el juego, deberemos rellenar todos los campos y ya podremos acceder al juego.

4- Una vez dentro del juego, podemos visualizar la foto relacionada con el juego, chatear con el resto de participantes, consultar los records o jugar directamente, seleccionando A, B o C a la pregunta planteada. Cada pregunta vale 100 puntos y cada ronda tiene 50 preguntas.

5- Si deseamos salir del juego, con cerrar la ventana del navegador, es suficiente.

6- Si deseamos volver a entrar en el juego, con introducir el nick y el password será suficiente.

7- Del lado del administrador del sistema, podemos variar las fotografías ubicadas en el interior de la carpeta con el nombre del mismo juego; se debe respetar el formato del nombre de cada fotografía (foto(XX).jpeg donde XX debe ser el número de fotografía) y se debe respetar también el orden numérico.

8- Así mismo, dentro del archivo access puede consultar los usuarios registrados, modificar las preguntas o ver las conexiones efectuadas al sistema.

Archivos

Dentro de la carpeta TFC se encuentran las siguientes carpetas:

a- Archivos JAVA: aquí están los archivos JAVA originales y donde se puede modificar los datos para adaptarlos a nuevos juegos.

b- Archivos CLASS: archivos compilados y resultantes de compilar los archivos JAVA.

c- Herramientas: aquí se encuentra la base de datos que puede ser consultada y modificada y más tarde incorporada al juego. También el archivo comprimido JAR y la forma de realizarlo. También se incluye la versión de Java utilizada para la compilación. Por último, se ha incluido el servidor XAMPP por si seria necesario para jugar en red.

d- Carpeta Juego: Juego original de la Cuadrilla de Zuia.

e- Carpeta juego adaptado: Juego basado en la UOC, una rápida adaptación del juego original.

Código fuente

ClienteHandler.java

```
import java.net.*;
import java.io.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.util.Random;
import javax.swing.border.*;
import javax.swing.table.TableModel;
import javax.swing.event.TableModelEvent;
import javax.swing.JTable;
import javax.swing.table.AbstractTableModel;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumnModel;
import javax.swing.text.*;

////////////////////////////////////////////////////////////////////////////
///  CLASE CLIENTEHANDLER
//APPLET SECUNDARIO, PANTALLA DE JUEGO
////////////////////////////////////////////////////////////////////////////


public  class ClienteHandler extends JApplet implements Runnable{


    JPanel jPzonaIzquierda = new JPanel();
    JPanel jPzonaDerecha = new JPanel();
    JPanel jSPpreguntas= new JPanel();
    JPanel jPpanelFotos = new JPanel();
    JPanel jPpanelEstadisticas= new JPanel();
    JPanel jPpanelPreguntas=new JPanel();
    JPanel jPpanelBordePreguntas=new JPanel();
    JPanel jPpanelChat=new JPanel();
    JPanel jPpanelInputChat=new JPanel();
    JPanel jPRespuestas=new JPanel();
    JPanel jPBotones=new JPanel();
    JPanel jPOpciones=new JPanel();
    JPanel jPTituloPreguntas=new JPanel();
```

```
JPanel jPTitulo=new JPanel();
JPanel jPCategoria=new JPanel();
JPanel jPUsers= new JPanel();
JPanel jPPuntos= new JPanel();
JPanel jPpanelUsuarios= new JPanel();
JPanel jPanelEnviar = new JPanel();

JScrollPane jSPTextoChat = new JScrollPane();
JScrollPane jSPListaUsuarios=new JScrollPane();
JScrollPane jSPListaRecords=new JScrollPane();

MyTableModel myModel = new MyTableModel();
MyTableModel2 myModel2 = new MyTableModel2();

JTable tablaPuntuaciones = new JTable(myModel);
JTable tablaRecords = new JTable(myModel2);

JTextField jTFnumPreg= new JTextField(20);
JTextField jTFtitulo = new JTextField(15);
JTextField jTFcategoria= new JTextField(20);

JLabel jLFoto = new JLabel();

//////////////////////
////// TIPOS DE LETRA
//////////////////////
Font comic = new Font("Comic Sans MS",Font.PLAIN,14 );
Font verdana=new Font("Verdana",Font.BOLD,14);

//////////////////////
// COLORES, tocando aqui podemos cambiar facilmente la apariencia del programa
//////////////////////


Color cFondoPantallaGeneral= new Color(255,255,255); //blanco
Color cFondoTitulo = new Color(173,3,11);
Color cTituloBorde= new Color(243,120,126);
Color cLineaBorde=new Color(0,0,0);
Color cBotonSeleccionado=new Color(255,255,255);
Color cBotonSeleccionadoRespuesta=new Color(5,255,5);
Color cFondoPantalla = new Color(255,0,0);
Color cLetrasChat= new Color(255,255,255);

//////////////////
```

```
   // BORDES
   ////////////////////
   Border bPorDefecto=jTFcategoria.getBorder();
   Border bBlanco=BorderFactory.createLineBorder(Color.white,2);
   Border bEspacio=  BorderFactory.createEmptyBorder(10,10,10,10);
   Border bEspacioFoto=  BorderFactory.createEmptyBorder(5,5,5,5);
   Border bUnEspacio=  BorderFactory.createEmptyBorder(0,5,0,5);
   Border bEspacioizdoderecho=  BorderFactory.createEmptyBorder(0,10,0,10);
   Border bLineaExterior=  BorderFactory.createLineBorder(cLineaBorde, 1);
   Border bLineaBlanca=  BorderFactory.createLineBorder(cLineaBorde, 1);
   Border bOpcionSeleccionadaPrevio=  BorderFactory.createLineBorder(cBotonSeleccionadoRespuesta, 1);
   Border bOpcionSeleccionada=BorderFactory.createCompoundBorder(bOpcionSeleccionadaPrevio,
bUnEspacio);
   Border bTituloChat= BorderFactory.createTitledBorder(bLineaBlanca, "Chat", TitledBorder.LEFT,
TitledBorder.ABOVE_TOP, verdana, cTituloBorde);
   Border bTituloLUsers= BorderFactory.createTitledBorder(bLineaBlanca, "Información
Records", TitledBorder.LEFT, TitledBorder.ABOVE_TOP, verdana, cTituloBorde);
   Border bTituloPreguntas= BorderFactory.createTitledBorder(bLineaBlanca, "Preguntas",
TitledBorder.LEFT, TitledBorder.ABOVE_TOP, verdana, cTituloBorde);
   Border bChat= BorderFactory.createCompoundBorder(bLineaExterior,bTituloChat);
   Border bPreguntas= BorderFactory.createCompoundBorder(bLineaExterior,bTituloPreguntas);
   Border bUsers= BorderFactory.createCompoundBorder(bLineaExterior,bTituloLUsers);

   ////////////////////////////
   // JTextPane, es una variante de jtextbox, permite cambiar el color de la letra pero
   // es mas complicado a la hora de gestionarlo.
   // solo se utilizan para las respuestas, ya que son las que necesitan colores.
   ////////////////////////////

  private Style estiloVerde,estiloOriginal;
  private JTextPane texto;

  private JTextPane creaEditor() {
    StyleContext sc = creaEstilos();
    DefaultStyledDocument doc = new DefaultStyledDocument( sc );

    return( texto = new JTextPane( doc ) );
  }

   private StyleContext creaEstilos() {
    StyleContext sc = new StyleContext();

    estiloVerde = sc.addStyle( null,null );
    estiloOriginal = sc.addStyle( null,null );
```

```
  StyleConstants.setBackground(estiloVerde,cFondoPantalla);
  StyleConstants.setBackground(estiloOriginal,cFondoPantalla);
  StyleConstants.setStrikeThrough(estiloOriginal,false);
  StyleConstants.setStrikeThrough(estiloVerde,false);
  StyleConstants.setForeground(estiloVerde,Color.yellow );
  StyleConstants.setForeground(estiloOriginal,cLetrasChat);
  StyleConstants.setFontFamily(estiloVerde,"Comic Sans MS");
  StyleConstants.setFontSize(estiloVerde,11);
  StyleConstants.setFontFamily(estiloOriginal,"Comic Sans MS");
  StyleConstants.setFontSize(estiloOriginal,11);

  return( sc );
}


StyleContext sc = creaEstilos();
DefaultStyledDocument doc = new DefaultStyledDocument( sc );

JTextPane jTFrespA = creaEditor();
JTextPane jTFrespB = creaEditor();
JTextPane jTFrespC = creaEditor();
JTextPane jTFpregunta= creaEditor();

JTextField jTFresultado1=new JTextField(100);
JTextField jTFresultado2=new JTextField(100);
JTextField jTFresultado3=new JTextField(100);



TextField input = new TextField();
JTextArea JTsalidaChat = new JTextArea();
JTextArea JTnicks= new JTextArea();
JTextArea JTpuntos= new JTextArea();
JButton jBEnviar = new JButton();
JButton jBRespA=new JButton();
JButton jBRespB=new JButton();
JButton jBRespC=new JButton();
DataInputStream i;
DataOutputStream o;


int numRespCorrecta;
String nick;
```

```
int nFoto; //numero de fotografias
String sNFich; //nombre de la fotografia, ej : "foto"
String sExt; //extension de la fotografia, ej: ".jpg"
String tematica; //tematica del juego, permite diferenciar los mensajes que corresponden
                 //a uno u otro juego.
int acerto=0;


//puntero a la clase TRIVIAL
//permitira invocar a metodos de dicha clase, para acceder a fotografias
Trivial triv;


//////////////////
protected Thread listener;
/// nueva instancia


 public void init(InputStream i, OutputStream o, String Username,int numFoto, String sNombreFich, String
sExtension, String tipoJuego, Trivial tr) {

   this.i = new DataInputStream (new BufferedInputStream (i));
   this.o = new DataOutputStream (new BufferedOutputStream (o));
   try  {

    //recogemos los parametros que vienen de la ventana de login.
     nick=Username;
     nFoto=numFoto;
     sNFich=sNombreFich;
     sExt=sExtension;
     triv=tr;
     tematica=tipoJuego;
     jbInit();
   }
   catch (Exception e) {
   }
   input.requestFocus ();
   listener = new Thread (this);
   listener.start ();
 }

 // inicializamos el estado de la nueva instancia
 private void jbInit() throws Exception {
```

```
this.getContentPane().setBackground(cFondoPantallaGeneral);
input.requestFocus ();

 ////////////////////////
//bordes para el chat
////////////////////////
jPpanelChat.setBorder(bChat);
jPpanelChat.setBackground(cFondoTitulo);
jSPTextoChat.setBackground(cFondoTitulo);
JTsalidaChat.setBorder(bEspacio);
JTsalidaChat.setBackground(cFondoPantalla);
JTsalidaChat.setForeground(cLetrasChat);
JTsalidaChat.setHighlighter(null);

 ////////////////////////
//bordes y apariencia para el panel de preguntas y el de respuestas
////////////////////////

jPpanelBordePreguntas.setBorder(bPreguntas);
jPpanelBordePreguntas.setBackground(cFondoTitulo);
jPpanelPreguntas.setBorder(bEspacio);
jPpanelPreguntas.setBackground(cFondoPantalla);
jTFcategoria.setBackground(cFondoPantalla);
jTFcategoria.setForeground(cLetrasChat);
jTFcategoria.setHighlighter(null);

jTFnumPreg.setBackground(cFondoPantalla);
jTFnumPreg.setForeground(cLetrasChat);
jTFnumPreg.setHighlighter(null);
jTFnumPreg.setHorizontalAlignment(JTextField.RIGHT);

jPRespuestas.setBackground(cFondoPantalla);
jPBotones.setBackground(cFondoPantalla);
jPOpciones.setBackground(cFondoPantalla);


//////////////////
//borde para las fotos
//////////////////
jPpanelFotos.setBorder(bEspacioFoto);
jPpanelFotos.setBackground(cFondoTitulo);


////////////////////////
```

```
////borde y apariencia para el panel de usuarios y puntuaciones y records
//////////////////////////

tablaPuntuaciones.setFont(comic);
tablaPuntuaciones.setBackground(cFondoPantalla);
tablaPuntuaciones.setForeground(cLetrasChat);
tablaPuntuaciones.setSelectionBackground(cFondoPantalla);
tablaPuntuaciones.setSelectionForeground(cLetrasChat);

tablaRecords.setFont(comic);
tablaRecords.setBackground(cFondoPantalla);
tablaRecords.setForeground(cLetrasChat);
tablaRecords.setSelectionBackground(cFondoPantalla);
tablaRecords.setSelectionForeground(cLetrasChat);


 ////////////////////////////////////////
/// TIPOS DE LETRA Y COLORES
////////////////////////////////////////

jPpanelEstadisticas.setBorder(bUsers);
jPpanelEstadisticas.setBackground(cFondoTitulo);
jSPListaUsuarios.setBackground(cFondoTitulo);
jSPListaRecords.setBackground(cFondoTitulo);


JTsalidaChat.setFont(comic);
JTnicks.setFont(comic);
JTpuntos.setFont(comic);
jTFcategoria.setFont(comic);
jBRespA.setFont(comic);
jBRespB.setFont(comic);
jBRespC.setFont(comic);
jBEnviar.setFont(comic);

JTnicks.setText("");
JTnicks.setEditable(false);
JTpuntos.setText("");
JTpuntos.setEditable(false);
JTsalidaChat.setEditable(false);
jTFtitulo.setFont(new Font("Dialog", 1, 40));
jTFnumPreg.setEditable(false);
jTFnumPreg.setText("");
```

```
jTFpregunta.setEditable(false);

jTFcategoria.setEditable(false);
jTFcategoria.setText("Espere mientras finaliza la pregunta en curso...");
jTFrespA.setCharacterAttributes( estiloOriginal,true );
jTFrespA.setEditable(false);
jTFrespA.setBackground(cFondoPantalla);
jTFrespA.setBorder(bPorDefecto);
jTFrespA.setText("");

jTFrespB.setCharacterAttributes( estiloOriginal,true );
jTFrespB.setEditable(false);
jTFrespB.setBackground(cFondoPantalla);
jTFrespB.setBorder(bPorDefecto);
jTFrespB.setText("");

jTFrespC.setCharacterAttributes( estiloOriginal,true );
jTFrespC.setEditable(false);
jTFrespC.setBackground(cFondoPantalla);
jTFrespC.setBorder(bPorDefecto);
jTFrespC.setText("");

jTFpregunta.setCharacterAttributes( estiloOriginal,true );
jTFpregunta.setEditable(false);
jTFpregunta.setBackground(cFondoPantalla);
jTFpregunta.setBorder(bPorDefecto);
jTFpregunta.setText("");

jBEnviar.setText("Enviar");
jBRespA.setFocusable(false);
jBRespA.setText("A");
jBRespB.setFocusable(false);
jBRespB.setText("B");
jBRespC.setFocusable(false);
jBRespC.setText("C");

 //los botones para responder estan inicialmente desactivados...
jBRespA.setEnabled(false);
jBRespB.setEnabled(false);
jBRespC.setEnabled(false);


////////////
//el contenedor principal es de tipo boxlayout
```

```
    //inserta sub-contenedores de izquierda a derecha
    //////////////
    this.getContentPane().setLayout(new BoxLayout((this.getContentPane()),BoxLayout.X_AXIS));
    this.getContentPane().add(Box.createRigidArea(new Dimension(13,0)));
    this.getContentPane().add(jPzonaIzquierda); //459px
    this.getContentPane().add(Box.createRigidArea(new Dimension(25,0)));
    this.getContentPane().add(jPzonaDerecha);  //500px



//////////////////
//zona izquierda
//////////////////

// contenedor boxlayout de arriba a abajo,
    jPzonaIzquierda.setLayout(new BoxLayout(jPzonaIzquierda, BoxLayout.Y_AXIS));
    jPzonaIzquierda.setPreferredSize(new Dimension(423, 645));
    jPzonaIzquierda.setMinimumSize(new Dimension(423, 645));
    jPzonaIzquierda.setMaximumSize(new Dimension(423, 645));
    jPzonaIzquierda.setBackground(cFondoPantallaGeneral);

    jPzonaIzquierda.add(jPpanelFotos);
    jPzonaIzquierda.add(Box.createRigidArea(new Dimension(0,15)));
    jPzonaIzquierda.add(jPpanelEstadisticas);

///////////////////
//zona izquierda superior, imagenes
///////////////////
    jPpanelFotos.setLayout(new BoxLayout(jPpanelFotos,BoxLayout.X_AXIS));
    jPpanelFotos.setPreferredSize(new Dimension(410, 410));
    jPpanelFotos.setMinimumSize(new Dimension(410, 410));
    jPpanelFotos.setMaximumSize(new Dimension(410, 410));
    jPpanelFotos.add(jLFoto);

///////////////
//zona izquierda inferior, estadisticas
///////////////

    jPpanelEstadisticas.setLayout(new BoxLayout(jPpanelEstadisticas, BoxLayout.X_AXIS));
    jPpanelEstadisticas.setPreferredSize(new Dimension(410, 120));
    jPpanelEstadisticas.setMinimumSize(new Dimension(410, 120));
    jPpanelEstadisticas.setMaximumSize(new Dimension(410, 120));
    jPpanelEstadisticas.add(jSPListaUsuarios);
    jPpanelEstadisticas.add(Box.createRigidArea(new Dimension(10,0)));
```


```
    jPpanelEstadisticas.add(jSPListaRecords);

//scroll pane para el listado de usuarios
    jSPListaUsuarios.setPreferredSize(new Dimension(240, 90));
    jSPListaUsuarios.setMinimumSize(new Dimension(240, 90));
    jSPListaUsuarios.setMaximumSize(new Dimension(240, 90));

jSPListaUsuarios.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
    jSPListaUsuarios.getViewport().add(tablaPuntuaciones, null);
    tablaPuntuaciones.setPreferredScrollableViewportSize(new Dimension(240, 90));

//scroll pane para el listado de records
    jSPListaRecords.setPreferredSize(new Dimension(150, 90));
    jSPListaRecords.setMinimumSize(new Dimension(150, 90));
    jSPListaRecords.setMaximumSize(new Dimension(150, 90));

jSPListaRecords.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
    jSPListaRecords.getViewport().add(tablaRecords, null);

    tablaRecords.setPreferredScrollableViewportSize(new Dimension(150, 90));



///////////////
//zona derecha
///////////////

// contenedor boxlayout de arriba a abajo
    jPzonaDerecha.setLayout(new BoxLayout(jPzonaDerecha, BoxLayout.Y_AXIS));
    jPzonaDerecha.setPreferredSize(new Dimension(500, 645));
    jPzonaDerecha.setMinimumSize(new Dimension(500, 645));
    jPzonaDerecha.setMaximumSize(new Dimension(500, 645));
    jPzonaDerecha.setBackground(cFondoPantallaGeneral);

    jPzonaDerecha.add(jPpanelBordePreguntas);
    jPzonaDerecha.add(Box.createRigidArea(new Dimension(0,15)));
    jPzonaDerecha.add(jPpanelChat);

///////////////
//zona de preguntas, superior derecha
///////////////

//contenedor para poder poner el borde con el titulo
  jPpanelBordePreguntas.setLayout(new BoxLayout(jPpanelBordePreguntas, BoxLayout.Y_AXIS));
```

```
 jPpanelBordePreguntas.setPreferredSize(new Dimension(500, 300));
 jPpanelBordePreguntas.setMinimumSize(new Dimension(500, 300));
 jPpanelBordePreguntas.setMaximumSize(new Dimension(500, 300));
   jPpanelBordePreguntas.add(jPpanelPreguntas);

///////////////
// contenedor de que apila de arriba a abajo
///////////////

 jPpanelPreguntas.setLayout(new BoxLayout(jPpanelPreguntas, BoxLayout.Y_AXIS));
 jPpanelPreguntas.setPreferredSize(new Dimension(490, 270));
 jPpanelPreguntas.setMinimumSize(new Dimension(490, 270));
 jPpanelPreguntas.setMaximumSize(new Dimension(490, 270));
 jPpanelPreguntas.add(jPTituloPreguntas);
 jPpanelPreguntas.add(Box.createRigidArea(new Dimension(0,10)));
 jPpanelPreguntas.add(jPTitulo);
 jPpanelPreguntas.add(Box.createRigidArea(new Dimension(0,10)));
 jPpanelPreguntas.add(jPRespuestas);
 jPpanelPreguntas.add(Box.createRigidArea(new Dimension(0,25)));

/////////////////////////////
///zona de titulo preguntas
/// apila de izq a derecha
/////////////////////////////

 jPTituloPreguntas.setLayout(new BoxLayout(jPTituloPreguntas, BoxLayout.X_AXIS));
 jPTituloPreguntas.setPreferredSize(new Dimension(490, 24));
 jPTituloPreguntas.setMinimumSize(new Dimension(490, 24));
 jPTituloPreguntas.setMaximumSize(new Dimension(490, 24));

 jPTituloPreguntas.add(jPCategoria);
 jPTituloPreguntas.add(jTFnumPreg);

 // panel que contiene la pregunta, permite especificar la anchura del cuadro
 // del textpane "preguntas"
 jPTitulo.setLayout(new BoxLayout(jPTitulo, BoxLayout.X_AXIS));
 jPTitulo.setPreferredSize(new Dimension(500, 60));
 jPTitulo.setMinimumSize(new Dimension(500, 60));
 jPTitulo.setMaximumSize(new Dimension(500, 60));
 jPTitulo.add(jTFpregunta);

 //panel que contiene categoria
 jPCategoria.setLayout(new BoxLayout(jPCategoria, BoxLayout.X_AXIS));
 jPCategoria.setPreferredSize(new Dimension(370, 24));
```

```
  jPCategoria.setMinimumSize(new Dimension(370, 24));
  jPCategoria.setMaximumSize(new Dimension(370, 24));
  jPCategoria.add(jTFcategoria);



////////////////
// zona de categoria respuestas, superior derecha inferior
////////////////
// contenedor que apila de izq a derecha

  jPRespuestas.setLayout(new BoxLayout(jPRespuestas, BoxLayout.X_AXIS));
  jPRespuestas.setPreferredSize(new Dimension(500, 150));
  jPRespuestas.setMinimumSize(new Dimension(500, 150));
  jPRespuestas.setMaximumSize(new Dimension(500, 150));
  jPRespuestas.add(jPBotones);
  jPRespuestas.add(jPOpciones);

  ///////////////
// zona de botones A, B, C, zona superior derecha superior, tercera linea  parte izquierda
///////////////
//contenedor de arriba a abajo

  jPBotones.setLayout(new BoxLayout(jPBotones, BoxLayout.Y_AXIS));
  jPBotones.setPreferredSize(new Dimension(50, 150));
  jPBotones.setMinimumSize(new Dimension(50, 150));
  jPBotones.setMaximumSize(new Dimension(50, 150));
  jPBotones.add(Box.createRigidArea(new Dimension(0,15)));
  jPBotones.add(jBRespA);
  jPBotones.add(Box.createRigidArea(new Dimension(0,25)));
  jPBotones.add(jBRespB);
  jPBotones.add(Box.createRigidArea(new Dimension(0,25)));
  jPBotones.add(jBRespC);



////////////////
// zona de frases de las respuestas, zona superior derecha superior tercera linea parte derecha
////////////////
//contenedor de arriba a abajo

  jPOpciones.setLayout(new BoxLayout(jPOpciones, BoxLayout.Y_AXIS));
  jPOpciones.setPreferredSize(new Dimension(418, 150));
  jPOpciones.setMinimumSize(new Dimension(418, 150));
```

```
 jPOpciones.setMaximumSize(new Dimension(418, 150));
 jPOpciones.add(jTFrespA);
 jPOpciones.add(Box.createRigidArea(new Dimension(0,5)));
 jPOpciones.add(jTFrespB);
 jPOpciones.add(Box.createRigidArea(new Dimension(0,5)));
 jPOpciones.add(jTFrespC);

///////////////
// zona de chat, centro inferior, apila de arriba a abajo
///////////////

   jPpanelChat.setLayout(new BoxLayout(jPpanelChat, BoxLayout.Y_AXIS));
   jPpanelChat.setPreferredSize(new Dimension(500, 230));
   jPpanelChat.setMinimumSize(new Dimension(500, 230));
   jPpanelChat.setMaximumSize(new Dimension(500, 230));

   jPpanelChat.add(jSPTextoChat);
   jPpanelChat.add(jPpanelInputChat);

 //scroll pane para el chat
   jSPTextoChat.setPreferredSize(new Dimension(500, 180));
   jSPTextoChat.setMinimumSize(new Dimension(500, 180));
   jSPTextoChat.setMaximumSize(new Dimension(500, 180));
   jSPTextoChat.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
   jSPTextoChat.getViewport().add(JTsalidaChat, null);


 //subcontenedor para la entrada de info del chat
 // apila de izq a derecha
   jPpanelInputChat.setLayout(new BoxLayout(jPpanelInputChat, BoxLayout.X_AXIS));
   jPpanelInputChat.setPreferredSize(new Dimension(500, 20));
   jPpanelInputChat.setMinimumSize(new Dimension(500, 20));
   jPpanelInputChat.setMaximumSize(new Dimension(500, 20));
   jPpanelInputChat.add(jBEnviar);
   jPpanelInputChat.add(input);


   mostrarFotografia(); //cargamos una primera fotografia en lo que llegan las preguntas...

   //LE PASAMOS LA INFO AL SERVIDOR DE QUE HEMOS ENTRADO AL JUEGO...
   notificarEntrada();


   ////////////////////////////
```

```
   /// EVENTO QUE SE PRODUCE AL PULSAR EL BOTON DE ENVIAR...
   ////////////////////////////

   jBEnviar.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
          tratarEnvioChat();
     }
   });
//-----------------------------------------------------


   //evento que se produce al hacer "return" sobre el campo input
   //permite enviar texto sin tener que darle al boton enviar, simplemente haciendo enter
   input.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
          tratarEnvioChat();
     }
   });



  ////////////////////////////
   /// EVENTO DE CLICK SOBRE EL BOTON RESPUESTA_A
   ////////////////////////////
  jBRespA.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {

         jBRespA.setBackground(cBotonSeleccionado);
         jTFrespA.setBorder(bBlanco);
         enviarResultado(1);
          //solo se puede resp 1 vez...
          desactivarBotones();

     }
   });


  ////////////////////////////
   /// EVENTO DE CLICK SOBRE EL BOTON RESPUESTA_B
   ////////////////////////////
  jBRespB.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {

         jBRespB.setBackground(cBotonSeleccionado);
```

```
        jTFrespB.setBorder(bBlanco);
        enviarResultado(2);
      //solo se puede resp 1 vez...
         desactivarBotones();

     }
   });


  ////////////////////////////
   /// EVENTO DE CLICK SOBRE EL BOTON RESPUESTA_C
   ////////////////////////////
  jBRespC.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {

        jBRespC.setBackground(cBotonSeleccionado);
        jTFrespC.setBorder(bBlanco);
        enviarResultado(3);

         desactivarBotones();

     }
   });

  }
////////////////////////////////////
// Enviamos al servidor una trama con el mensaje de chat
// TRAMA: idchat, tematica, separador, mensaje de chat
////////////////////////////////////////

void tratarEnvioChat(){
          try {

          // si el campo a enviar esta vacio no se envia nada
          //lo sabemos porque el string resultante tiene longitud 0

          if ((((input.getText()).length()))>0) {
          o.writeUTF (ConstantesCliente.ID_TRAMA_CHAT + tematica + ConstantesCliente.SEPARADOR +
"<" + nick + "> "+ input.getText());
          o.flush ();
          }


          }
```

```
            catch (IOException ex) {
            ex.printStackTrace();
            }
            input.setText ("");


}

 ////////////////////////////////////////////
 // PROCESA LA RESPUESTA Y ENVIA EL RESULT AL SERVER
 //trama: idespecial + tematica + SEPARADOR + nick + SEPARADOR + 1 o 0 segun acierte o no
 ////////////////////////////////////////////

 public void enviarResultado(int result){
    try{
  String respuesta;

            //miramos si ha acertado la respuesta
            if (numRespCorrecta==result) {
               acerto=1;
            }
            else {
               acerto=0;
            }
   respuesta=ConstantesCliente.ID_TRAMA_PREG_RESP + tematica + ConstantesCliente.SEPARADOR +
nick + ConstantesCliente.SEPARADOR + acerto;
   o.writeUTF (respuesta);
   o.flush ();

     }

            catch (IOException ex) {
            ex.printStackTrace();

            }
 }


 ////////////////////////////////////////////
 /// ENVIAMOS UNA TRAMA DE AVISO CONFORME HEMOS ENTRADO
 //caracteres generados con ALT+129,150,158,166
 ////////////////////////////////////////////
 public void notificarEntrada(){
    try{
```

```
    String resp;
    resp=ConstantesCliente.ID_TRAMA_ENTRADA + tematica + ConstantesCliente.SEPARADOR + nick;
    o.writeUTF (resp);
    o.flush ();
     }
    catch (IOException ex) {
          ex.printStackTrace();

          }
}

////////////////////////////////////////////////
  /// ENVIAMOS UNA TRAMA DE AVISO CONFORME HEMOS SALIDO
   //caracteres generados con ALT+129,150,158,163
  ////////////////////////////////////////////////
  public void notificarSalida(){
     try{
    String resp;
    resp=ConstantesCliente.ID_TRAMA_SALIDA + tematica + ConstantesCliente.SEPARADOR + nick;
    o.writeUTF (resp);
    o.flush ();
    }
     catch (IOException ex) {
          ex.printStackTrace();

          }
}


  // funcion que desactiva los botones de respuesta
   public void desactivarBotones(){

          jBRespA.setEnabled(false);
          jBRespB.setEnabled(false);
          jBRespC.setEnabled(false);
    }

////////////////////////////////////////////////////////////////////////////
  ////////// aqui se analiza la trama recibida
////////////////////////////////////////////////////////////////////////////
  public void run () {
    String s;
    int tipoTrama;
```

```
int ignorarRespuestas=1;  //permite que al entrar en medio de una preg no nos diga que fallamos
try {
  while (true) {
     // leemos de tipo UFT, es mejor ya que informa de longitud enviada
    String line = i.readUTF ();
    tipoTrama=analizarTrama(line);

    switch(tipoTrama){
       case 1: //llegan preguntas
       tratarInformacionPreguntasRespuestas(line);
       acerto=0; //reseteamos el aviso de acierto... ya que es una nueva pregunta
       ignorarRespuestas=0;
          break;

       case 2: // llega una trama con informacion del numero de usuarios conectados
          tratarInformacionUsuarios(line.substring(4));

          break;

       case 3: //llega un listado con las puntuaciones de la ultima pregunta....
          //si aun no nos ha llegado ninguna pregunta con anterioridad, ignoramos las resps
          if(ignorarRespuestas==0){
          tratarInformacionPuntuaciones(line.substring(4));
          }
          break;

       default:  //por defecto tratamos la trama como un mensaje de chat o de info
             tratarInformacionChat(line);
             break;


  } //switch
} //while

} //try
  catch (IOException ex) {
  ex.printStackTrace ();
} finally {
  listener = null;
  validate ();
  try {
    o.close ();
  } catch (IOException ex) {
    ex.printStackTrace ();
```

```
    }
   }
  }

  ////////////////////////////////////////////
  // funcion que se ocupa de los mensajes de chat y tambien de los mensajes
  // de notificacion de entrada y salida al juego.
  ////////////////////////////////////////////

  void tratarInformacionChat(String linea){

//comprobamos si el mensaje de chat lo ha originado un jugador
//de nuestro mismo juego, lo sabemos comparando la tematica

    int posicionInicial,posicionFinal=0;

    //buscamos el separador, para ver donde acaba la categoria
    posicionFinal=linea.indexOf(ConstantesCliente.SEPARADOR,0);
    //sacamos la pregunta por pantalla

if (linea.substring(0,posicionFinal).equals(tematica)){
      JTsalidaChat.append(linea.substring(posicionFinal+1) + "\n");
}
  }



  void tratarInformacionPuntuaciones(String linea){

      String reesp, SrespCorrecta;


       //DESACTIVAMOS LOS BOTONES PARA RESPONDER YA QUE ESTAMOS TRABAJANDO
CON LAS RESPUESTAS
      // Y HA EXPIRADO EL TIEMPO PARA PREGUNTAR...
           jBRespA.setEnabled(false);
           jBRespB.setEnabled(false);
           jBRespC.setEnabled(false);

      switch(numRespCorrecta){
      case 1:  //respCorrecta A
              jTFrespA.setBorder(bOpcionSeleccionada);
              jTFrespA.setCharacterAttributes( estiloVerde,true );
              jTFrespA.setText(jTFrespA.getText());
```

```
                break;
        case 2:// respCorrecta B
                jTFrespB.setBorder(bOpcionSeleccionada);
                jTFrespB.setCharacterAttributes( estiloVerde,true );
                jTFrespB.setText(jTFrespB.getText());
                break;
        case 3:// respCorrecta C
                jTFrespC.setBorder(bOpcionSeleccionada);
                jTFrespC.setCharacterAttributes( estiloVerde,true );
                jTFrespC.setText(jTFrespC.getText());
                break;
          default:SrespCorrecta="";
                break;
    }

    tratarInformacionUsuarios(linea);

}

////////////////////////////////////////////////////////////////
//TRAMA: idtrama tematica separador numtanda separador preg/resp
////////////////////////////////////////////////////////////////

void tratarInformacionPreguntasRespuestas (String linea){
    int posicionInicial,posicionFinal=0;
    String car;
    String pathFoto;

    //buscamos el separador, para ver donde acaba la tematica
        posicionFinal=linea.indexOf(ConstantesCliente.SEPARADOR,0);

        //si la trama es de la tematica de nuestro juego continuamos
        //en caso contrario ignoramos la trama
        if (tematica.equals(linea.substring(4,posicionFinal))){


    //buscamos el separador, para ver donde acaba la categoria
        posicionInicial=posicionFinal+1;
        posicionFinal=linea.indexOf(ConstantesCliente.SEPARADOR,posicionInicial);
        //sacamos la pregunta por pantalla

        jTFnumPreg.setText("Pregunta " + linea.substring(posicionInicial,posicionFinal) + "/" +
ConstantesCliente.PREGUNTASTANDA);
```

```
        //buscamos el separador, para ver donde acaba la pregunta
        posicionInicial=posicionFinal+1;
        posicionFinal=linea.indexOf(ConstantesCliente.SEPARADOR,posicionInicial);
        jTFcategoria.setText(linea.substring(posicionInicial,posicionFinal));

        //buscamos el separador, para ver donde acaba la pregunta
        posicionInicial=posicionFinal+1;
        posicionFinal=linea.indexOf(ConstantesCliente.SEPARADOR,posicionInicial);
        jTFpregunta.setText(" " + linea.substring(posicionInicial,posicionFinal));


        //buscamos el separador, para ver donde acaba la respA
        posicionInicial=posicionFinal+1;
        posicionFinal=linea.indexOf(ConstantesCliente.SEPARADOR,posicionInicial);
        jTFrespA.setText(" " + linea.substring(posicionInicial,posicionFinal));
          //pregB
        posicionInicial=posicionFinal+1;
        posicionFinal=linea.indexOf(ConstantesCliente.SEPARADOR,posicionInicial);
        jTFrespB.setText(" " + linea.substring(posicionInicial,posicionFinal));
          //pregC
        posicionInicial=posicionFinal+1;
        posicionFinal=linea.indexOf(ConstantesCliente.SEPARADOR,posicionInicial);
        jTFrespC.setText(" " + linea.substring(posicionInicial,posicionFinal));

        //respbuena (1,2 o 3)
        posicionInicial=posicionFinal+1;
        //recojemos el numero de la resp correcta
       car=(linea.substring(posicionInicial));
       numRespCorrecta=Integer.parseInt(car);


      //ACTIVAMOS LOS BOTONES PARA RESPONDER YA QUE HEMOS RECIBIDO UNA
PREGUNTA...
      //restauramos el color y refrescamos para que surta efecto

        jBRespA.setBackground(null);
        jBRespA.setEnabled(true);
        jBRespB.setBackground(null);
        jBRespB.setEnabled(true);
        jBRespC.setBackground(null);
        jBRespC.setEnabled(true);

        jTFrespA.setCharacterAttributes( estiloOriginal,true );
        jTFrespA.setText(jTFrespA.getText());
```

```
        jTFrespA.setBorder(bPorDefecto);
        jTFrespB.setCharacterAttributes( estiloOriginal,true );
        jTFrespB.setText(jTFrespB.getText());
        jTFrespB.setBorder(bPorDefecto);
        jTFrespC.setCharacterAttributes( estiloOriginal,true );
        jTFrespC.setText(jTFrespC.getText());
        jTFrespC.setBorder(bPorDefecto);

        mostrarFotografia(); // mostramos una nueva fotografia por pantalla.
        }// end if
}

////////////////////////////////////////////////
///// MOSTRAR FOTOGRAFIA
///// obtenemos una nueva foto y la mostramos////
////////////////////////////////////////////////
void mostrarFotografia(){

 String fich;

  //montamos el fichero de la fotografia a mostrar a partir de los parametros
  //que recibimos (cabecera, numero de foto, extension)
  //el numero de foto que generamos aleatoriamente es de 0 a nfoto-1 por eso le
  //sumamos 1, para obtener un num entre 1 i nfoto

 fich=  sNFich + (numAzar(nFoto)+1) + sExt;

  //pedimos al applet principal que descargue una nueva imagen del servidor
  //lo ha de hacer necesariamente el applet principal "trivial", en caso
  //contrario se produce un error.

  triv.pedirFotografia(fich); //guarda una nueva fotografia en "iconologin"
  jLFoto.setIcon(triv.iconologin); //mostramos la nueva fotografia
}


 ////////////////////////////////////
// funcion que devuelve un numero aleatorio entre 0 y "numero"-1
////////////////////////////////////
int numAzar ( int numero){
int result;
Random rnd = new Random();
result=rnd.nextInt(numero);
return result;
```

```
}


////////////////////////////////
/// PROCESA LA INFORMACION DE LOS USUARIOS ENTRANTES
// separa la subtrama de puntuaciones de la de records.
//IMPORTANTE, TRAMA:
//idtramanumusers|nick#puntuacion|nick#puntuacion....          nickrecord    SEPARADORRECORDS
puntmax..

////////////////////////////////
void tratarInformacionUsuarios(String cadena){
 int posicionInicial=0, posicionFinal=0;
 String cadenaPuntos, cadenaRecords,tem;

 //decodificamos la tematica de la trama
 posicionFinal=cadena.indexOf(ConstantesCliente.SEPARADOR,posicionInicial);
 tem=cadena.substring(posicionInicial,posicionFinal);
 posicionInicial=posicionFinal+1;

 ////////////////////////
 // solo atendemos la trama si es de nuestra tematica, en caso contrario se ignora.
 ////////////////////////


 if (tem.equals(tematica)){

        //decodificamos el resto de la trama.
        posicionFinal=cadena.indexOf(ConstantesCliente.SEPARADORRECORDS,posicionInicial);
        cadenaPuntos=cadena.substring(posicionInicial,posicionFinal);
        cadenaRecords=cadena.substring(posicionFinal+1,cadena.length());

        tratarPuntuaciones(cadenaPuntos);
        tratarRecords(cadenaRecords);

 }//if

}


////////////////////////////////////////////////////////////////
// tratamos la subtrama de puntuaciones, decodificamos la subtrama
```

```
// ordenamos de mayor a menor puntuacion e insertamos en la jtable
////////////////////////////////////////////////////////////////////
void tratarPuntuaciones(String cadenaPuntos){
int posicionInicial=0, posicionFinal=0, fila=0;

   /// tratamos puntos

 posicionInicial=0;
 posicionFinal=0;

 ////////////////
 //con los dos vectores formamos un array bidimensional para representar una tabla
 //cada fila consta de 3 columnas: nombre, puntuacion, tresp.
 ///////////////

Vector vOrdenarDatos= new Vector();
Vector vAux = null;

//primer usuario

 //username
 posicionFinal=cadenaPuntos.indexOf(ConstantesCliente.SEPARADOR,posicionInicial);
 vAux= new Vector(); //creamos una nueva estructura, a razon de 1 por fila (usuario)
 vAux.add(cadenaPuntos.substring(posicionInicial,posicionFinal)); //añado username

 //puntuacion
 posicionInicial=posicionFinal+1;
 posicionFinal=cadenaPuntos.indexOf(ConstantesCliente.SEPARADORTRESP,posicionInicial);
 vAux.add(cadenaPuntos.substring(posicionInicial,posicionFinal)); //añado puntuacion

 //tresp
 posicionInicial=posicionFinal+1;
 posicionFinal=cadenaPuntos.indexOf(ConstantesCliente.SEPARADOR2,posicionInicial);

 vAux.add(cadenaPuntos.substring(posicionInicial,posicionFinal)); //añado tiempo de respuesta


 vOrdenarDatos.clear(); //inicializo vector
 vOrdenarDatos.add(vAux); //añado fila

 fila++;

  //resto de usuarios
```

```
    while (posicionFinal<(cadenaPuntos.length()-1)){
    posicionInicial=posicionFinal+1;
    posicionFinal=cadenaPuntos.indexOf(ConstantesCliente.SEPARADOR,posicionInicial);
    vAux= new Vector();
    vAux.add(cadenaPuntos.substring(posicionInicial,posicionFinal));

    //puntuacion
    posicionInicial=posicionFinal+1;
    posicionFinal=cadenaPuntos.indexOf(ConstantesCliente.SEPARADORTRESP,posicionInicial);
    vAux.add(cadenaPuntos.substring(posicionInicial,posicionFinal)); //añado puntuacion

    //tresp
    posicionInicial=posicionFinal+1;
    posicionFinal=cadenaPuntos.indexOf(ConstantesCliente.SEPARADOR2,posicionInicial);
    vAux.add(cadenaPuntos.substring(posicionInicial,posicionFinal)); //añado tiempo de respuesta

    vOrdenarDatos.add(vAux);

    fila++;
}


//llamamos a la clase que ordena, especificando que ordene por columna 1 (puntuacion)
//y de mayor a menor (orden descendente)

Collections.sort(vOrdenarDatos, new ColumnSorter(1, false));

// con la informacio ya ordenada, la pasamos de los 2 vectores a una jtable
// para mostrar por pantalla
pasarAVector(vOrdenarDatos);

//rellenamos con "" hasta el final, para dejar en blanco el resto de posiciones

    while (fila<ConstantesCliente.MAXUSERS){
    tablaPuntuaciones.setValueAt("",fila,0);
    tablaPuntuaciones.setValueAt("",fila,1);
    tablaPuntuaciones.setValueAt("",fila,2);
    fila++;
    }


//liberamos recursos
vOrdenarDatos=null;
```

```
vAux=null;
}


////////////////////////////////////////////////
// tratamos la subtrama de records, decodificandola e insertandola en la jtable
////////////////////////////////////////////////
void tratarRecords(String cadena){
   int posicionInicial=0, posicionFinal=0,fila=0;

  //la informacion viene directa de la base de datos y se ha extraido ya ordenada
  //asi que al contrario que con los datos de puntuaciones, no hace falta ordenar
  //e insertamos directamente en la jtable
  //TRAMA QUE NOS LLEGA AQUI: usuario SEPARADOR puntrecord SEPRECORD usuario....

 fila=0;
 while (posicionFinal<(cadena.length()-1)){

     posicionFinal=cadena.indexOf(ConstantesCliente.SEPARADOR,posicionInicial);
    tablaRecords.setValueAt((cadena.substring(posicionInicial,posicionFinal)),fila,0);

     //puntuacion
     posicionInicial=posicionFinal+1;
     posicionFinal=cadena.indexOf(ConstantesCliente.SEPARADORRECORDS,posicionInicial);
     tablaRecords.setValueAt((cadena.substring(posicionInicial,posicionFinal)),fila,1);

     posicionInicial=posicionFinal+1;

     fila++;
    }//while
}



////////////////////////////
// recorre el vector de vectores y va añadiendo la info en la jtable
////////////////////////////
 public void pasarAVector(Vector vOrdenarDatos){
    int i,j;
 i=0;
 j=0;

 while (i<vOrdenarDatos.size()){
```

```
    j=0;
    while (j<((Vector)vOrdenarDatos.elementAt(i)).size()){
     tablaPuntuaciones.setValueAt(((Vector)vOrdenarDatos.elementAt(i)).elementAt(j),i,j);
    j++;
   }
 i++;
 }
}



 ////////////////////
 // analiza las tramas entrantes identificando su tipo
 ////////////////////
    int analizarTrama (String mensaje ){

if (mensaje.length()>=4) { //los msjs de chat no llevan trama y pueden medir menos de 4caracts

            //TRAMA que contiene la pregunta y las respuestas
        if (mensaje.substring(0,4).equals(ConstantesCliente.ID_TRAMA_PREG_RESP)){
            return (1);
        }
        else {//trama que contiene listado de ususarios
            if(mensaje.substring(0,4).equals(ConstantesCliente.ID_TRAMA_INFOUSERS)){
              return (2);
             }


            else {//respuesta a la pregunta
              if (mensaje.substring(0,4).equals(ConstantesCliente.ID_TRAMA_PUNTUACIONES)){
                 return (3);
              }
               return(-1);
              }
        }

}
 else return (-1);

 }



}
```

```
/////////////////////////////////
/// clase que permite crear la tabla de puntuaciones de usuario
//////////////////////////////////
class MyTableModel extends AbstractTableModel {
        final String[] columnNames = {"Usuario",
                        "Puntuación",
                        "T. resp (seg.)"
                        };
         //ponemos 100 filas, si hubiera mas usuarios habria que añadir mas
         Object[][] data = { {"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},
         {"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},
         {"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},
         {"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},
         {"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},
         {"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},
         {"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},
         {"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},
         {"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},
         {"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""},{"","",""}
        };


        //únicamente retornamos el numero de elementos del
        //array de los nombres de las columnas
        public int getColumnCount() {
            return columnNames.length;
        }

        //retormanos el numero de elementos
        //del array de datos
        public int getRowCount() {
            return data.length;
        }

        //retornamos el elemento indicado
        public String getColumnName(int col) {
            return columnNames[col];
        }

        //y lo mismo para las celdas
        public Object getValueAt(int row, int col) {
            return data[row][col];
```

```
    }

    public void setValueAt(Object value, int row, int col) {
        data[row][col] = value;
        fireTableCellUpdated(row, col);
    }

}



////////////////////////////////
/// clase que permite crear la tabla de records
////////////////////////////////
class MyTableModel2 extends AbstractTableModel {
    final String[] columnNames = {"Usuario",
                                  "Puntuación",
                                  };
     //ponemos 11 filas, si queremos mas records hay que añadir mas
     Object[][] data = { {"",""},{"",""},{"",""},{"",""},{"",""},{"",""},{"",""},{"",""},{"",""},{"",""},{"",""}
    };


    //únicamente retornamos el numero de elementos del
    //array de los nombres de las columnas
    public int getColumnCount() {
        return columnNames.length;
    }

    //retormanos el numero de elementos
    //del array de datos
    public int getRowCount() {
        return data.length;
    }

    //retornamos el elemento indicado
    public String getColumnName(int col) {
        return columnNames[col];
    }

    //y lo mismo para las celdas
    public Object getValueAt(int row, int col) {
        return data[row][col];
    }
```

```
    public void setValueAt(Object value, int row, int col) {
        data[row][col] = value;
        fireTableCellUpdated(row, col);
    }

}
```

ColumnSorter.java

```
import java.lang.Comparable.*;
import java.util.*;


/////////////////////////////////////////////////////
// Clase que permite ordenar un jTable
/////////////////////////////////////////////////////

// This comparator is used to sort vectors of data
    public class ColumnSorter implements Comparator {
        int colIndex;
        boolean ascending;
        ColumnSorter(int colIndex, boolean ascending) {
            this.colIndex = colIndex;
            this.ascending = ascending;
        }
        public int compare(Object a, Object b) {
            Vector v1 = (Vector)a;
            Vector v2 = (Vector)b;
            Object o1 = v1.get(colIndex);
            Object o2 = v2.get(colIndex);

            // Treat empty strains like nulls
            if (o1 instanceof String && ((String)o1).length() == 0) {
                o1 = null;
            }
            if (o2 instanceof String && ((String)o2).length() == 0) {
                o2 = null;
            }

            // Sort nulls so they appear last, regardless
            // of sort order
            if (o1 == null && o2 == null) {
                return 0;
            } else if (o1 == null) {
                return 1;
            } else if (o2 == null) {
                return -1;
            } else if (o1 instanceof Comparable) {
                if (ascending) {
                    return ((Comparable)o1).compareTo(o2);
```


```
            } else {
                return ((Comparable)o2).compareTo(o1);
            }
        } else {
            if (ascending) {
                return o1.toString().compareTo(o2.toString());
            } else {
                return o2.toString().compareTo(o1.toString());
            }
        }
    }
}
```

ConstantesCliente.java

```
public interface ConstantesCliente {

   ////////////////////////////////////////////////
   // OPCIONES DE CONFIGURACION DEL JUEGO
   ////////////////////////////////////////////////

   final int PREGUNTASTANDA=50;  //permite configurar de cuantas preguntas queremos la ronda...
   final int MAXUSERS=100;  //constante para jtable, en principio no tocar
   final int MINCARACTERES=4; //long minima de los campos de user y pass

   ////////////////////////////////////////////////
   // IDENTIFICADORES DE TRAMAS, no tocar
   ////////////////////////////////////////////////

   final String ID_TRAMA_AUTENTIFICACION="üû×ô";
   final String ID_TRAMA_ENTRADA="üû×ª";
   final String ID_TRAMA_SALIDA="üû×ú";
   final String ID_TRAMA_PREG_RESP="üû×º";
   final String ID_TRAMA_CHAT="üû×x";
   final String ID_TRAMA_INFOUSERS="üû×É";
   final String ID_TRAMA_PUNTUACIONES="üû×á";
   final String SEPARADOR="|";
   final String SEPARADOR2="#";
   final String SEPARADORRECORDS="â";
   final String SEPARADORTRESP="¨";
   final String ID_TRAMA_CIERRE_CLIENTE="üû×?";
   final String ID_TRAMA_REGISTRO="üû×ÿ";
   final String ID_TRAMA_REGISTRO_CORRECTO="üû×¦";
   final String ID_TRAMA_PERMISOPARAJUGAR="üû×æ";
   final String ID_TRAMA_SALIDASINNICK="ûx×æ";



}
```

ConstantesServidor.java

```
import java.util.*;
public interface ConstantesServidor {

   ////////////////////////////////////////////////
   // OPCIONES DE CONFIGURACION DEL JUEGO
   ////////////////////////////////////////////////


   // PERMITE AÑADIR O ELIMINAR TIPOS DE JUEGO
   // el parametro "directorio_fotos" que recibe el applet debera ser uno de los
   // nombres listados en este parametro
   // en el access tendremos que tener, para cada tipo de juego:
   // p.ej para la tematica "deportes"
   // tabla preguntas_deportes
   // en la tabla "usuarios" un campo de nombre "maxdeportes"

   final String TEMATICAS_JUEGO="zuia";

   //numero de juegos que tenemos
   final int NUMERO_JUEGOS=1;
   final int PREGUNTASTANDA=50;  //permite configurar de cuantas preguntas queremos la ronda...
   final int NUMMAXACERTANTES=5; //numero de maximos acertantes,RECORDS, que mostraremos en
el juego...
   final int INCREMENTOPUNTOS=100; //permite configurar cuantos puntos concedemos por preg
acertada
   final String RUTA_BD="C:/Zuia/Castellano/online/TrivialOnLine.mdb"; //path del fichero de access
   final String USER_BD="Administrador";  //usuario de la BD
   final String PASS_BD="trivialonline";  //contraseña de la BD

   //PERMITE ALARGAR O ACORTAR EL TIEMPO ENTRE PREGS
   // la diferencia entre ambas constantes sera el tiempo que hay para ver los resultados
   // es decir, el tiempo en el que se ve la respuestas correcta por pantalla.
   final int tiempoEntrePreguntas=18000;
   final int tres=12000;

   // PUERTO QUE UTILIZARA EL SERVIDOR, CAMBIARLO SI YA ESTA EN USO
   static final int PUERTO=4322;
```

```
 ////////////////////////////////////////////////
// IDENTIFICADORES DE TRAMAS, no tocar
////////////////////////////////////////////////
  final String ID_TRAMA_AUTENTIFICACION="üû×ô";
  final String ID_TRAMA_REGISTRO="üû×ÿ";
  final String ID_TRAMA_REGISTRO_CORRECTO="üû×¦";
  final String ID_TRAMA_PERMISOPARAJUGAR="üû×æ";
  final String ID_TRAMA_ENTRADA="üû×ª";
  final String ID_TRAMA_SALIDA="üû×ú";
  final String ID_TRAMA_SALIDASINNICK="ûx×æ";
  final String ID_TRAMA_PREG_RESP="üû×º";
  final String ID_TRAMA_CHAT="üû×x";
  final String ID_TRAMA_INFOUSERS="üû×É";
  final String ID_TRAMA_PUNTUACIONES="üû×á";
  final String SEPARADOR="|";
  final String SEPARADOR2="#";
  final String SEPARADORTRESP="¨";
  final String SEPARADORRECORDS="â";

}
```

ServidorHandler.java

```
import java.net.*;
import java.io.*;
import java.util.*;
import java.util.Random;
import java.sql.*;
import java.text.*;


public class ServidorHandler extends Thread {


  protected Socket s;
  protected DataInputStream i;
  protected DataOutputStream o;
  ArrayList y = new ArrayList();
  int genpreg, tiempo,tiempores;
  ArrayList coleccionPreguntas= new ArrayList();  //listado de preguntas
  ArrayList listadoJuegos= new ArrayList(); //listado de los nombres de los juegos

  int numeroTanda=1; ///numero de pregunta
  String acum;

  public ServidorHandler(Socket s, ArrayList x, int generadorPreguntas, int tiempoEntrePreguntas,int tres, ArrayList coleccionP, ArrayList listadoJueg) throws IOException {
    this.s = s;
    i = new DataInputStream (new BufferedInputStream (s.getInputStream ()));
    o = new DataOutputStream (new BufferedOutputStream (s.getOutputStream ()));
    y = x;
    genpreg=generadorPreguntas;
    tiempo=tiempoEntrePreguntas;
    tiempores=tres;
    coleccionPreguntas=coleccionP;
    listadoJuegos=listadoJueg;

  }

  // VARIABLES COMPARTIDAS POR LOS THREADS
  // UN THREAD PUEDE MODIFICARLAS Y EL RESULTADO ES VISIBLE POR EL RESTO DE THREADS
  protected static ArrayList handlers = new ArrayList ();
  protected static ArrayList aPuntuacionesUsuarios = new ArrayList();
```

```
protected static long tiempoInicial;
protected static ArrayList aNombremaximosAcertantes=new ArrayList(); //tabla bidimensional
protected static ArrayList aPuntmaximosAcertantes=new ArrayList(); //tabla bidimensional
protected static int iPrecord[]= new int[ConstantesServidor.NUMERO_JUEGOS]; //lista unidimensional
protected static int numRecords; //numero de personas que hay en la tabla

public void run () {
  String name = s.getInetAddress ().toString ();
  String Listado="";
  String nombre="";
  int tipoTrama,contador=0,encontrado=0;
  boolean Salir=false;


  try {

     //dimensionamos las tablas, habra tantas filas como juegos/canales
     iniTablaPuntuaciones();
     iniTablaNombreMaximos();
     iniTablaPuntMaximos();

     ////////////////////////////////
     //// Unicamente la primera replica del servidor generara preguntas y tramas de puntuacion...
     ////////////////////////////////
     if (genpreg==1) {

        //mandamos recargar las tablas de records de todos los juegos
        for(int cont=0;(cont<aNombremaximosAcertantes.size());cont++){
          cargarRecords((String)listadoJuegos.get(cont));
        }


       Timer timer= new Timer();
        timer.scheduleAtFixedRate(new Preguntas(),tiempo,tiempo);
     }



     for (int i=0; i < y.size() ; i++){
                  Listado=Listado+y.get(i)+"\n";
     }


     handlers.add(this);
```

```
        System.out.println(handlers.size() + " usuario/s on-line");
       Salir=false;
      while (Salir==false) {
         String msg = i.readUTF ();

//////////////////////////////////////////////
 //mirar si es trama especial en ese caso devolveremos trama con total de puntos.
//////////////////////////////////////////////
        tipoTrama=analizarTrama(msg);

        switch(tipoTrama){
           case 1:  //entra un usuario , mantenemos el arraylist de #usuarios...
                // enviamos al cliente la info actualizada de los usuarios
              tratarEntradaUsuario(msg.substring(4));
              break;

           case 2: //salida de usuario, mantenemos el arraylist de #usuarios...
              // enviamos al cliente la info actualizada de los usuarios
              tratarSalidaUsuario(msg.substring(4));
              Salir=true; // salimos de la instancia del servidor que atendia al cliente
              break;
           case 3:
              tratarRespuesta(msg.substring(4));
              break;
           case 4: //mensaje de chat
              broadcast (msg.substring(4));
              break;
           case 5: //mensaje de autentificacion, contiene user y passs
              comprobarAutentificacion(msg.substring(4));
              break;
           case 6: //mensaje de registro, contiene user, pass,nombre,mail
              realizarRegistro(msg.substring(4));
              break;
           case 7: //mensaje de salida sin haber llegado a hacer login
               Salir=true; // salimos de la instancia del servidor que atendia al cliente
                break;
           default: //aqui no deberia entrar, si entra es trama erronea
              System.out.println("ERROR: TRAMA ERRONEA");
              break;
        }

      }
    } catch (IOException ex) {
      ex.printStackTrace ();
```

```
  } finally {

   //tareas a realizar cuando el cliente cierra el juego
    handlers.remove(this);
    y.remove(name);
    try {
     //cerramos el socket correspondiente.
     i.close();
     o.close();
     s.close();
     System.out.println(handlers.size() + " usuario/s on-line");

    } catch (IOException ex) {
     ex.printStackTrace();
    }
   }
 }

//////////////////////////////////////
// Creamos un array de hashmaps para guardar los pares jugador, puntuacion
//////////////////////////////////////
void iniTablaPuntuaciones(){
for (int i=0;i<ConstantesServidor.NUMERO_JUEGOS;i++){
   HashMap hAux= new HashMap();
   aPuntuacionesUsuarios.add(hAux);
}

}


//////////////////////////////////////
// Creamos un array de arrays para guardar los pares jugador, puntuacion
//////////////////////////////////////
void iniTablaNombreMaximos(){
for (int i=0;i<ConstantesServidor.NUMERO_JUEGOS;i++){
   ArrayList aAux= new ArrayList();
   aAux.clear();
   aNombremaximosAcertantes.add(aAux);
}

}


 //////////////////////////////////////
```

```
// Creamos un array de arrays para guardar los pares jugador, puntuacion
//////////////////////////////////////
void iniTablaPuntMaximos(){
for (int i=0;i<ConstantesServidor.NUMERO_JUEGOS;i++){
    ArrayList aAux= new ArrayList();
    aAux.clear();
    aPuntmaximosAcertantes.add(aAux);
}

}



//////////////////////////////////////////
// Tratamos la trama que recibe el servidor cuando entra un nuevo jugador
// tipo_de_juego SEPARADOR nick
//////////////////////////////////////////

 void tratarEntradaUsuario(String cadena){

   String tematica,nick;
   int posicionInicial=0,posicionFinal=0;
   int numJuego;
   //buscamos el separador, para identificar la tematica y el nick
   posicionFinal=cadena.indexOf(ConstantesServidor.SEPARADOR,posicionInicial);

   tematica=cadena.substring(posicionInicial,posicionFinal);
   nick=cadena.substring(posicionFinal+1);
   broadcast (cadena+" ha entrado en el juego");
   numJuego=listadoJuegos.indexOf(tematica); //buscamos la posicion de la tematica del jugador entrante
   ((HashMap)aPuntuacionesUsuarios.get(numJuego)).put(nick,"0"                                                    +
ConstantesServidor.SEPARADORTRESP); //añadimos user con punt0


   //enviamos una trama con la info actualizada de los usuarios para la tematica que ha
   //cambiado
      enviarListadoUsuarios(tematica);
 }

 //////////////////////////////////////////
// Tratamos la trama que recibe el servidor cuando entra un nuevo jugador
// tipo_de_juego SEPARADOR nick
//////////////////////////////////////////
```

```
void tratarSalidaUsuario(String cadena){
    String tematica,nick;
  int posicionInicial=0,posicionFinal=0;
  int numJuego;
  //buscamos el separador, para identificar la tematica y el nick
  posicionFinal=cadena.indexOf(ConstantesServidor.SEPARADOR,posicionInicial);

  tematica=cadena.substring(posicionInicial,posicionFinal);
  nick=cadena.substring(posicionFinal+1);
  broadcast (cadena+" ha abandonado el juego");
  numJuego=listadoJuegos.indexOf(tematica); //buscamos la posicion de la tematica del jugador entrante
  ((HashMap)aPuntuacionesUsuarios.get(numJuego)).remove(nick);

  //enviamos una trama con la info actualizada de los usuarios para la tematica que ha
  //cambiado
     enviarListadoUsuarios(tematica);

}

//////////////////////////////////////////////////////
// funcion que trata la trama de respuestas
//trama esperada: tematica + SEPARADOR + nick + SEPARADOR + 1 o 0 segun acierte o no
//////////////////////////////////////////////////////
void tratarRespuesta(String cadena){

int posicionInicial=0, posicionFinal=0;
String usuario;
String sOldPunt;
String tem;
int oldPunt, acierto;

  // decodificamos la tematica/canal
  posicionFinal=cadena.indexOf(ConstantesServidor.SEPARADOR,posicionInicial);
  tem=(cadena.substring(posicionInicial,posicionFinal));
  posicionInicial=posicionFinal+1;

  // decodificamos el usuario
  posicionFinal=cadena.indexOf(ConstantesServidor.SEPARADOR,posicionInicial);
  usuario=(cadena.substring(posicionInicial,posicionFinal));
  posicionInicial=posicionFinal+1;


  //acierto o no
  acierto=Integer.parseInt(cadena.substring(posicionInicial,cadena.length()));
```

```
posicionInicial=0;
posicionFinal=0;

/// calculamos el tiempo de respuesta, mirando cuanto tiempo
// ha transcurrido desde que se envio la pregunta
java.util.Date d2 = new java.util.Date();
long tiempoFinal = d2.getTime();
long diferencia = tiempoFinal - tiempoInicial;
long segundos=diferencia/1000;
long milisegundos= diferencia- (segundos*1000);
String tiempoDeRespuesta=(segundos + ","+ milisegundos);


//buscamos el nick, siempre lo encontrara ya que solo pueden entrar
//usuarios registrados
//lo buscamos en la fila adecuada, de acuerdo a la id de tematica recibida

if(((HashMap)aPuntuacionesUsuarios.get(listadoJuegos.indexOf(tem))).containsKey(usuario)){
   //el nick ya existe
   //cojemos la puntuacion antigua
   sOldPunt=(String)((HashMap)aPuntuacionesUsuarios.get(listadoJuegos.indexOf(tem))).get(usuario);

   //decodificandola ya que viene junto con
   //el tiempo de respuesta anterior
   posicionFinal=sOldPunt.indexOf(ConstantesServidor.SEPARADORTRESP,posicionInicial);
   sOldPunt=(sOldPunt.substring(posicionInicial,posicionFinal));

   if (acierto==1){ //si ha habido acierto incrementamos la puntuacion
   //paso a int y incrementamos
   oldPunt=Integer.parseInt(sOldPunt)+ConstantesServidor.INCREMENTOPUNTOS;
   //mirar si es record y actualizar la BD
   actualizarRecords(tem,usuario, oldPunt);
   }
   else{
   oldPunt=Integer.parseInt(sOldPunt);
   }
    //pasamos de int a string
    sOldPunt= String.valueOf( oldPunt);
    //concatenamos el tiempo de respuesta del usuario en cuestion
    sOldPunt=sOldPunt + ConstantesServidor.SEPARADORTRESP + tiempoDeRespuesta;
    //actualizo la hasttable con la nueva puntuacion
   ((HashMap)aPuntuacionesUsuarios.get(listadoJuegos.indexOf(tem))).put(usuario,sOldPunt);
```

```
}

 }

 ////////////////////////////////////////////
 // cuando un usuario acierta una pregunta se ejecuta esta funcion
 // comprueba si ha hecho record y si lo ha hecho actualiza la bd...
 // cada usuario puede hacer records en diferentes juegos...
 ////////////////////////////////////////////
 void actualizarRecords(String tem, String user, int punt){
  String Srecord;
  int posicion, iPuntosUser=0, irec;
  String sPuntosUser;


     //cojemos la puntuacion que es record y es mas baja
     //como esta ordenado de + a - sera la ultima de la lista.
    //si ha superado el record mas bajo hay que actualizar la BD de records
    //y tambien si aun no esta llena
   irec=(iPrecord[listadoJuegos.indexOf(tem)]);
    if ( (punt>irec) || (numRecords<ConstantesServidor.NUMMAXACERTANTES) ) {

       //conectamos a la BD
      try{
       String myDB,insertar="";
       Connection con;

       Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
       myDB="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ="+ConstantesServidor.RUTA_BD;

con=DriverManager.getConnection(myDB,ConstantesServidor.USER_BD,ConstantesServidor.PASS_BD);
       Statement lectura2 = con.createStatement();
       Statement lectura=con.createStatement();
       //consultamos los puntos que tiene el usuario que ha hecho el record,
       //podria ser que ya tenga un record mayor en cuyo caso no actualizamos

       ResultSet records= lectura.executeQuery("select max" + tem +" from usuarios where nick='" + user + "';");
       records.next(); //avanzamos el EOF
       sPuntosUser=records.getString("max" + tem);
       iPuntosUser=Integer.parseInt(sPuntosUser);

       if (punt>iPuntosUser){ //solo actualizamos si era su mejor record
```

```
     insertar= "UPDATE USUARIOS SET max" + tem +" =" + punt + " WHERE NICK='" + user + "';";
     lectura2.executeUpdate(insertar);
     }
    con.close();
   } //try
    catch(Exception ex){
     System.out.println("Tabla records de la BD no accesible.");

  }//catch

   if (punt>iPuntosUser){  //refrescamos solo si ha habido cambios
   //ahora refrescamos las estructuras que contienen los records
    cargarRecords(tem);  //refrescamos la tabla de records que se ha modificado...
   }

  }//if


}// fin actualizar Records



//////////////////////////
/// funcion que recoje tematica/canal, user y pass,nombre,mail y mira si existe en la BD y
//  lo da de alta en la misma, comunicandolo al cliente.
//////////////////////////
void realizarRegistro(String linea){
Connection con;
int posicionInicial=0, posicionFinal=0;
String myDB,sNick,sPass,sNombre,sEmail,trama,insertar,tem;

  //buscamos la tematica/juego

  posicionFinal=linea.indexOf(ConstantesServidor.SEPARADOR,0);
  tem=linea.substring(0,posicionFinal);
  posicionInicial=posicionFinal+1;

   //decodificamos la trama, separando user y pass

   posicionFinal=linea.indexOf(ConstantesServidor.SEPARADOR,posicionInicial);

  sNick=linea.substring(posicionInicial,posicionFinal);

  //buscamos el separador
```

```
   posicionInicial=posicionFinal+1;
   posicionFinal=linea.indexOf(ConstantesServidor.SEPARADOR,posicionInicial);
   sPass=linea.substring(posicionInicial,posicionFinal);

   //buscamos el separador
   posicionInicial=posicionFinal+1;
   posicionFinal=linea.indexOf(ConstantesServidor.SEPARADOR,posicionInicial);
   sNombre=linea.substring(posicionInicial,posicionFinal);

   //buscamos el separador
   posicionInicial=posicionFinal+1;
   posicionFinal=linea.length();
   sEmail=linea.substring(posicionInicial,posicionFinal);

// comprobamos si el usuario ha introducido algun caracter invalido, en ese caso se descarta el ingreso
if (caracterValido(sNick) && caracterValido(sPass) && caracterValido(sNombre) && caracterValido(sEmail)
){

String consulta="SELECT * FROM usuarios WHERE nick='" + sNick + "'";

 try
   {

  //conectamos con la BD y checkeamos para ver si user y pass existen
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

  myDB="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ="+ConstantesServidor.RUTA_BD;
  con=DriverManager.getConnection(myDB,ConstantesServidor.USER_BD,ConstantesServidor.PASS_BD);
  Statement lectura = con.createStatement();
  ResultSet resultado = lectura.executeQuery(consulta);
  if (resultado.next()){
   // se ha encontrado el user por lo tanto no se puede registrar este usuario
   trama=ConstantesServidor.ID_TRAMA_REGISTRO_CORRECTO+"0";
  }//if
  else {  //dar de alta al usuario en la bd
   insertar="INSERT INTO usuarios(nick,password,nombre,email,max" + tem + ") VALUES ('" + sNick + "','"+
sPass+"','"+ sNombre+"','"+sEmail+"',0)";
   lectura.executeUpdate(insertar);
   trama=ConstantesServidor.ID_TRAMA_REGISTRO_CORRECTO+"1";

   }//else
  //enviamos info
  enviarDatos(trama);
  con.close();
```

```
 } //try

catch (ClassNotFoundException ex) {
System.err.println("Cannot find the DB driver classes.");
System.err.println(ex);
}

    catch (Exception ex)
    {
        System.out.println("Error al conectarse con la BD de usuarios");
    }

} // fin del if que verifica caracteres invalidos


else { // registro invalido ya que contiene datos incorrectos.
    trama=ConstantesServidor.ID_TRAMA_REGISTRO_CORRECTO+"0";
    enviarDatos(trama);
} //else

  }

  ////////////////////////////////////////////////
  // verifica que los datos de entrada no contengan datos invalidos
  // evitando asi posibles vulnerabilidades.
  ////////////////////////////////////////////////
  boolean caracterValido(String cadena){
   if    (    (cadena.indexOf("=")!=-1)    ||    (cadena.indexOf("<")!=-1)    ||    (cadena.indexOf(">")!=-1)    ||
(cadena.indexOf("'")!=-1) || (cadena.indexOf("/")!=-1) || (cadena.indexOf("!")!=-1) )  {
       return false;
   }
   else {
       return true;
   }
  }

  ////////////////////////
  /// funcion que recoje user y pass y mira si existe en la BD y lo comunica al cliente.
  ////////////////////////
  void comprobarAutentificacion(String linea){
  Connection con;
  int posicionInicial=0, posicionFinal=0;
  String myDB,sNick,sPass,trama,tem;
```

```
    //decodificamos la tematica
    posicionFinal=linea.indexOf(ConstantesServidor.SEPARADOR,0);
    tem=linea.substring(0,posicionFinal);
    posicionInicial=posicionFinal+1;
    //decodificamos la trama, separando user y pass

    posicionFinal=linea.indexOf(ConstantesServidor.SEPARADOR,posicionInicial);
   //sacamos la pregunta por pantalla
   sNick=linea.substring(posicionInicial,posicionFinal);

   //buscamos el separador
   posicionInicial=posicionFinal+1;
   posicionFinal=linea.length();
   sPass=linea.substring(posicionInicial,posicionFinal);

  // comprobamos si el usuario ha introducido algun caracter invalido, en ese caso se descarta el ingreso
if (caracterValido(sNick) && caracterValido(sPass) ){


   String consulta="SELECT * FROM usuarios WHERE nick='" + sNick + "' AND password='" + sPass + "'";

try
   {
  //conectamos con la BD y checkeamos para ver si user y pass existen
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
 myDB="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ="+ConstantesServidor.RUTA_BD;
con=DriverManager.getConnection(myDB,ConstantesServidor.USER_BD,ConstantesServidor.PASS_BD);
 Statement lectura = con.createStatement();
 ResultSet resultado = lectura.executeQuery(consulta);

 if (resultado.next()){
 // se ha encontrado el user y el pass es correcto


  //miramos que no este jugando, si ya esta jugando no puede entrar 2 veces
  if(((HashMap)aPuntuacionesUsuarios.get(listadoJuegos.indexOf(tem))).containsKey(sNick)){
    trama=ConstantesServidor.ID_TRAMA_PERMISOPARAJUGAR + "2";
  }
  else{
  // enviamos trama al cliente con la info
  trama=ConstantesServidor.ID_TRAMA_PERMISOPARAJUGAR + "1";
  }//else
 }//if
```

```
 else {
 // el user pass es incorrecto
 // informamos al cliente
 trama=ConstantesServidor.ID_TRAMA_PERMISOPARAJUGAR + "0";
 }

 enviarDatos(trama);
 con.close();
 }
   catch (Exception ex)
   {
      System.out.println("Error al conectarse con la BD de usuarios/comprobar autentificacion");
   }


} //end del if que comprueba caracteres invalidos.
else{
     trama=ConstantesServidor.ID_TRAMA_PERMISOPARAJUGAR + "0";
     enviarDatos(trama);
}
 }



 ///////////////
 // funcion que carga los records de la tabla access
 // busca los records en la tabla usuarios, campo "maxpuntuacion_tematica"
 ///////////////
 void cargarRecords(String campo){
   Connection con;
   String myDB,punt="0";
   int num=0,numAleatorio=0,iterador=1;
   try
   {

   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   myDB="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ="+ConstantesServidor.RUTA_BD;
   con=DriverManager.getConnection(myDB,ConstantesServidor.USER_BD,ConstantesServidor.PASS_BD);
   Statement lectura2 = con.createStatement();
   ResultSet records= lectura2.executeQuery("select nick, max" + campo +" from usuarios order by max" +
campo +" desc");

   //borramos la fila de los arrays antes de añadir
   ((ArrayList)aNombremaximosAcertantes.get(listadoJuegos.indexOf(campo))).clear();
```

```
   ((ArrayList)aPuntmaximosAcertantes.get(listadoJuegos.indexOf(campo))).clear();

   //recorremos hasta que se acaben los maximos acertantes o lleguemos al num maximo de
    // maximos acertantes
    numRecords=0;
    while( (records.next()) && (iterador<=ConstantesServidor.NUMMAXACERTANTES) ){
        //si la puntuacion es nula sacamos 0 por pantalla
        punt=records.getString("max"+campo);
       if (punt==null) {
           punt="0";
       }

((ArrayList)aNombremaximosAcertantes.get(listadoJuegos.indexOf(campo))).add(records.getString("nick"));
       ((ArrayList)aPuntmaximosAcertantes.get(listadoJuegos.indexOf(campo))).add(punt);
       iterador++;
       numRecords++;
    }
   //guardamos la puntuacion mas baja, si se supera esta puntuacion sabremos
  //que es un nuevo record...

       iPrecord[(listadoJuegos.indexOf(campo))]=Integer.parseInt(punt);
       con.close();

   } //try


   catch (ClassNotFoundException ex) {
   System.err.println("No se ha encontrado el driver para la BD.");
   System.err.println(ex);
}



   catch (Exception ex)
   {
   System.out.println("No se pudo realizar la conexion con la base de datos. "+ex.getMessage());
   System.exit(-1);
   }
 }
```

```
////////////////////////////////////
// funcion que devuelve un numero aleatorio entre 0 y "numero"
////////////////////////////////////
int numAzar ( int numero){
int result;
Random rnd = new Random();
result=rnd.nextInt(numero); //genera un numero aleatorio entre 0 y lo que recibe
return result;
}

//////////////////////////////
//envia a los clientes info actualizada de los usuarios que estan conectados.
//////////////////////////////
void enviarListadoUsuarios (String tem){
 String lista, convertir,listadoPuntuaciones="";
 int numJuego;
//montamos la trama
 //idtramanumusers|nick|nick|nick|
 int contador=0;
lista="";

      numJuego=listadoJuegos.indexOf(tem); //buscamos la posicion de la tematica del jugador entrante
        //aqui enviaremos una trama con todas las puntuaciones que hayamos recopilado.
      for( Iterator it2 = ((HashMap)aPuntuacionesUsuarios.get(numJuego)).keySet().iterator(); it2.hasNext();) {
            String s = (String)it2.next();
            String s1 = (String)((HashMap)aPuntuacionesUsuarios.get(numJuego)).get(s);
            listadoPuntuaciones=listadoPuntuaciones+(s + ConstantesServidor.SEPARADOR +
s1+ConstantesServidor.SEPARADOR2);
          }



   //recorremos el arraylist de maximos acertantes para añadirlo a la trama.

      Iterator it = ((ArrayList)aNombremaximosAcertantes.get(listadoJuegos.indexOf(tem))).iterator();
      Iterator it3= ((ArrayList)aPuntmaximosAcertantes.get(listadoJuegos.indexOf(tem))).iterator();
      while ( (it.hasNext()) && (it3.hasNext()) ){
      String snick=(String)(it.next());  //nick
      String spuntos=(String)(it3.next()); //punt maximos
```

```
        listadoPuntuaciones=listadoPuntuaciones + ConstantesServidor.SEPARADORRECORDS + snick + ConstantesServidor.SEPARADOR + spuntos;

        }

    broadcast(ConstantesServidor.ID_TRAMA_INFOUSERS+ tem + ConstantesServidor.SEPARADOR + listadoPuntuaciones + ConstantesServidor.SEPARADORRECORDS);

 }

 ////////////////////////////////////
 /// FUNCION QUE ANALIZA LAS TRAMAS QUE ENVIA EL CLIENTE
 // DEVUELVE UN ENTERO QUE INDICA TIPO DE TRAMA
 ////////////////////////////////////

 int analizarTrama (String mensaje ){

 if (mensaje.length()>=4) { //los msjs de chat no llevan trama y pueden medir menos de 4caracts

        //TRAMA DE ENTRADA
      if (mensaje.substring(0,4).equals(ConstantesServidor.ID_TRAMA_ENTRADA)){
          return (1);
      }
      else {//salida
        if(mensaje.substring(0,4).equals(ConstantesServidor.ID_TRAMA_SALIDA)){
          return (2);
         }
        else {//respuesta a la pregunta
          if (mensaje.substring(0,4).equals(ConstantesServidor.ID_TRAMA_PREG_RESP)){
              return (3);
          }
          else { // mensaje de chat
          if (mensaje.substring(0,4).equals(ConstantesServidor.ID_TRAMA_CHAT)){
              return (4);
          }
          else { //mensaje de autentificacion
          if (mensaje.substring(0,4).equals(ConstantesServidor.ID_TRAMA_AUTENTIFICACION)){
              return(5);
          }
```

```
            else { //mensaje de registo
            if (mensaje.substring(0,4).equals(ConstantesServidor.ID_TRAMA_REGISTRO)){
                return(6);
            }
            else { //mensaje de salida sin haber llegado a hacer login
            if (mensaje.substring(0,4).equals(ConstantesServidor.ID_TRAMA_SALIDASINNICK)){
                return(7);
            }

            else return(-1); //trama no reconocida
                    }
                }
              }
             }
            }
        }
 else return (-1);

}

 ////////////////////////////////////////////
 /// CLASE RESPUESTAS, ENVIA LAS PUNTUACIONES CUANDO EXPIRA EL TIEMPO PARA RESPONDER
 ////////////////////////////////////////////


 class Respuestas extends TimerTask{

    public void run(){

        //enviamos 1 trama para cada juego, el cliente leera la que le corresponda
        // e ignorara el resto
        for(int contador=0;(contador<coleccionPreguntas.size());contador++){
                enviarRespuestas((String)listadoJuegos.get(contador));
        }
          }//run

    void enviarRespuestas(String tem){
                String listadoPuntuaciones="";

        //preparamos la trama que contiene username , puntuacio, tiempo de respuesta
        for(                                Iterator                                it2                                =
```

```
((HashMap)aPuntuacionesUsuarios.get(listadoJuegos.indexOf(tem))).keySet().iterator(); it2.hasNext();) {

                String s = (String)it2.next();
                String s1 = (String)((HashMap)aPuntuacionesUsuarios.get(listadoJuegos.indexOf(tem))).get(s);
                listadoPuntuaciones=listadoPuntuaciones+(s + ConstantesServidor.SEPARADOR + s1+
ConstantesServidor.SEPARADOR2);
            }


      //recorremos el arraylist de maximos acertantes para añadirlo a la trama.
         Iterator it = ((ArrayList)aNombremaximosAcertantes.get(listadoJuegos.indexOf(tem))).iterator();
         Iterator it3= ((ArrayList)aPuntmaximosAcertantes.get(listadoJuegos.indexOf(tem))).iterator();

         while ( (it.hasNext()) && (it3.hasNext()) ){
         String snick=(String)(it.next());  //nick
         String spuntos=(String)(it3.next()); //punt maximos
         listadoPuntuaciones=listadoPuntuaciones + ConstantesServidor.SEPARADORRECORDS + snick +
ConstantesServidor.SEPARADOR + spuntos;

         }

      //envio de la trama
      broadcast(ConstantesServidor.ID_TRAMA_PUNTUACIONES + tem +
ConstantesServidor.SEPARADOR + listadoPuntuaciones + ConstantesServidor.SEPARADORRECORDS);


    }
    }
```

```
 ////////////////////////////////////
 //CLASE QUE SE EJECUTA CADA 12 SEGUNDOS.....
 //solo la ejecuta la 1a instancia o thread del servidor
 ////////////////////////////////////
class Preguntas extends TimerTask {
     public void run() {
       Timer timer= new Timer();
       int numeroAzar;
       String listadoPuntuaciones="";
```

```
        ////////
        // enviamos tramas de preguntas
        // enviamos las tramas de preguntas, el cliente ya mostrara la adecuada
        // segun su tematica (asi el servidor no se recarga innecesariamente)
        ///////

        for(int contador=0;(contador<coleccionPreguntas.size());contador++){

           // generamos un numero aleatorio, donde el min es 0, max numpreguntas
            numeroAzar=numAzar(((ArrayList)coleccionPreguntas.get(contador)).size());
            //enviamos la pregunta
            //TRAMA: idtrama tematica separador numtanda separador preg/resp
           broadcast(ConstantesServidor.ID_TRAMA_PREG_RESP + listadoJuegos.get(contador) +
ConstantesServidor.SEPARADOR +
numeroTanda+ConstantesServidor.SEPARADOR+((ArrayList)coleccionPreguntas.get(contador)).get(numero
Azar));
        }



        //cada vez que se genera una pregunta
        //encendemos un temporizador para que salte la respuesta
        //antes de la siguiente pregunta
        timer.schedule(new Respuestas(),tiempores);

        //si es la primera pregunta reseteamos las puntuaciones...
        //esto lo hacemos para todos los juegos

           if (numeroTanda==1){

        for (int ii=0;ii<aPuntuacionesUsuarios.size();ii++){

        for( Iterator it2 = ((HashMap)aPuntuacionesUsuarios.get(ii)).keySet().iterator(); it2.hasNext();) {

            String s = (String)it2.next();
            ((HashMap)aPuntuacionesUsuarios.get(ii)).put(s,"0"+ConstantesServidor.SEPARADORTRESP);
            String s1 = (String)((HashMap)aPuntuacionesUsuarios.get(ii)).get(s);

            listadoPuntuaciones=listadoPuntuaciones+(s + ConstantesServidor.SEPARADOR + s1+
ConstantesServidor.SEPARADOR2);
                //enviar trama actualizando las puntuaciones... basta como una de entrarsalidausers

            broadcast(ConstantesServidor.ID_TRAMA_INFOUSERS+ (String)listadoJuegos.get(ii) +
ConstantesServidor.SEPARADOR + listadoPuntuaciones + obtenerCadenaRecords(ii));
```

```
            }//for interior


        }//for exterior

        }//if

        else{ //si es una pregunta normal habra que resetear por lo menos los tiempos de
              //respuesta, ya que en caso contrario un usuario que no responda se quedara
              //con el t_resp anterior


            for (int ii=0;ii<aPuntuacionesUsuarios.size();ii++){

            for( Iterator it2 = ((HashMap)aPuntuacionesUsuarios.get(ii)).keySet().iterator(); it2.hasNext();) {

                String s = (String)it2.next();
                String s1 = (String)((HashMap)aPuntuacionesUsuarios.get(ii)).get(s);

                //decodificandola ya que viene junto con
                //el tiempo de respuesta anterior
                int posicionFinal=s1.indexOf(ConstantesServidor.SEPARADORTRESP,0);
                s1=s1.substring(0,posicionFinal);

               //añadimos otra vez la puntuacion pero sin el tresp
                ((HashMap)aPuntuacionesUsuarios.get(ii)).put(s,s1+ConstantesServidor.SEPARADORTRESP + "
- -");
            }//for interior
            }//for exterior

            }//else


       //incrementamos el numero de pregunta
                numeroTanda++;
            //miramos a ver si estamos en la misma tanda o no....
            if (numeroTanda==(ConstantesServidor.PREGUNTASTANDA+1)) {
                numeroTanda=1;}
            //poner puntuaciones a 0....


        /// ponemos a 0 el "cronometro" que nos permitira saber cuanto tarda en resp
    //  cada usuario
        java.util.Date d1 = new java.util.Date();
```

```
            tiempoInicial = d1.getTime();


        }//run


  ////////////////////////////////////////
  // para una fila (canal/tema) concreto, consulta las tablas de records
  // y devuelve la trama ya preparada para enviar al cliente.
  ////////////////////////////////////////
   String obtenerCadenaRecords(int fila){

        String records="";

        Iterator it = ((ArrayList)aNombremaximosAcertantes.get(fila)).iterator();
        Iterator it3= ((ArrayList)aPuntmaximosAcertantes.get(fila)).iterator();

        while ( (it.hasNext()) && (it3.hasNext()) ){
        String snick=(String)(it.next());  //nick
        String spuntos=(String)(it3.next()); //punt maximos
        records=records+ ConstantesServidor.SEPARADORRECORDS + snick +
ConstantesServidor.SEPARADOR + spuntos;
        }

        records=records +ConstantesServidor.SEPARADORRECORDS;
        return(records);
    }




    }



  // envia una trama a TODOS los clientes
  protected static void broadcast (String message) {
    synchronized (handlers) {

     for (Iterator iter = handlers.iterator(); iter.hasNext();) {
     ServidorHandler c = (ServidorHandler) iter.next();
     try {
          synchronized (c.o) {
            c.o.writeUTF (message);
```


```
        }
        c.o.flush ();
      }
    catch (IOException ex) {

      }
   } //for
   }
 }


 //envia una trama SOLO al cliente que le ha enviado la info.
 void enviarDatos(String message){
  try {
          ServidorHandler c= this;
        synchronized (c.o) {
          c.o.writeUTF (message);
        }
        c.o.flush ();
      } catch (IOException ex) {

      }
 }


}
```

ServidorTrivial.java

```
import java.net.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import java.text.*;

public class ServidorTrivial   {

    int numUsers=0;  ///variable global que cuenta los usuarios conectados
    int numPreguntas;
    int posicionInicial=0,posicionFinal=0;

    ArrayList coleccionPreguntas; // contiene las preguntas, es una tabla
    ArrayList listadoJuegos; // contiene los nombres de los juegos, util para donde esta la info en la tabla
    ArrayList aAux;  // utilizado para añadir filas de preguntas/respuestas a la tabla

    public ServidorTrivial(int port) throws IOException {

        ArrayList x= new ArrayList();
        int generadorpreguntas=1;

     try{

    ServerSocket server = new ServerSocket (port);
    System.out.println("Servidor activo, puerto " + port);

    //inicializamos la tabla (array de arrays)
    //en cada columna hay una pregunta/resp
    //cada fila es un juego (tematica/canal diferente)

    coleccionPreguntas=new ArrayList();
    listadoJuegos= new ArrayList();
    coleccionPreguntas.clear();

    while (true) {

      if (generadorpreguntas==1){
          System.out.println("Cargando preguntas de " + ConstantesServidor.RUTA_BD + "....");

        // leemos las diferentes tablas de preguntas
        // para ello consultamos las constantes de servidor que nos indican
```

```
        // cuantas tematicas o "canales" hay.
        // y lanzamos una carga para cada una de ellas

         for(int i=0;i<ConstantesServidor.NUMERO_JUEGOS;i++){
          if ((i+1)==ConstantesServidor.NUMERO_JUEGOS){ //en la ultima categoria no hay coma final
             posicionFinal=ConstantesServidor.TEMATICAS_JUEGO.length();
          }
          else{
             posicionFinal=ConstantesServidor.TEMATICAS_JUEGO.indexOf(",",posicionInicial);
          }

listadoJuegos.add(ConstantesServidor.TEMATICAS_JUEGO.substring(posicionInicial,posicionFinal));
          cargarPreguntas("preguntas_" +
ConstantesServidor.TEMATICAS_JUEGO.substring(posicionInicial,posicionFinal));
          posicionInicial=posicionFinal+1;
        }
         System.out.println("Ha finalizado la carga de preguntas. Se han cargado " +
ConstantesServidor.NUMERO_JUEGOS + " juegos");
    }

    Socket client = server.accept ();
    x.add(client.getInetAddress ());

    actualizarEstadisticas(client);

    ///////////////
    /// solo el primer "ServidorHandler" tendra el flag generadorpreguntas a 1
    ///////////////

    if (generadorpreguntas==1) {
     ServidorHandler c = new ServidorHandler
(client,x,generadorpreguntas,ConstantesServidor.tiempoEntrePreguntas,ConstantesServidor.tres,coleccionPreg
untas,listadoJuegos);
     c.start ();
    }
    else{
       ServidorHandler c = new ServidorHandler
(client,x,generadorpreguntas,ConstantesServidor.tiempoEntrePreguntas,ConstantesServidor.tres,null,listadoJu
egos);
     c.start ();
    }
    generadorpreguntas=0;
```

```
   }
    }
    catch(Exception ex){
       System.out.println("Error: ya hay una instancia de Trivial ejecutandose o bien el puerto "+
ConstantesServidor.PUERTO+ " esta ocupado por otra aplicacion");
    }
 }

 ////////////////////////////////////////
 // funcion que guarda info de los usuarios contectados en la BD access
 // ip, dia, mes, hora...
 ////////////////////////////////////////

  public void actualizarEstadisticas(Socket client){
   String myDB,insertar;
    Connection con;

   try{

     //conectamos con la BD y checkeamos para ver si user y pass existen
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    myDB="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ="+ConstantesServidor.RUTA_BD;

con=DriverManager.getConnection(myDB,ConstantesServidor.USER_BD,ConstantesServidor.PASS_BD);
    Statement lectura = con.createStatement();

    //preparamos fecha u hora para la insercion en la BD
    java.util.Date fechaActual= new java.util.Date();

    SimpleDateFormat formatoDia= new SimpleDateFormat("dd");
    SimpleDateFormat formatoMes= new SimpleDateFormat("MM");
    SimpleDateFormat formatoAño= new SimpleDateFormat("yyyy");
    SimpleDateFormat formatoHora= new SimpleDateFormat("H:mm");
    String hora=formatoHora.format(fechaActual);
    String dia=formatoDia.format(fechaActual);
    String mes=formatoMes.format(fechaActual);
    String año=formatoAño.format(fechaActual);

     insertar= "INSERT INTO Estadisticas(ip,dia,mes,año,hora) VALUES ('" +
client.getInetAddress().getHostName()+ "'," + dia+"','"+mes+"','"+año + "','"+hora+ "')";

     lectura.executeUpdate(insertar);
     con.close();
```

```
  }
  catch(Exception ex){
     System.out.println("Tabla estadisticas de la BD no accesible.");

  }
}

public static void main (String args[]) throws IOException {
  new ServidorTrivial (ConstantesServidor.PUERTO);
}


///////////////
// funcion que carga las preguntas de la tabla access
// mediante una funcion que genera un aleatorio "desordenamos" las respuestas
// para que no siempre sea la C la correcta
// ademas recoje tambien los nombres de los maximos acertantes.
///////////////
void cargarPreguntas(String tabla){
  Connection con;
  String myDB,trama,primera,segunda,tercera,numRespCorrecta,cad,punt="";
  int num=0,numAleatorio=0,iterador=1;
  long i=0;
  try
  {
  primera="";
  segunda="";
  tercera="";
  numRespCorrecta="";
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  myDB="jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ="+ConstantesServidor.RUTA_BD;
  con=DriverManager.getConnection(myDB,ConstantesServidor.USER_BD,ConstantesServidor.PASS_BD);
  Statement lectura = con.createStatement();

  ResultSet resultado = lectura.executeQuery("select * from " + tabla);

  aAux= new ArrayList();

  System.out.println("Cargando tabla " + tabla);

  while (resultado.next())
  {
    // la resp correcta siempre es la C, las vamos a "desordenar" para que vaya cambiando
```

```
//hacemos una espera entre cada generacion de numero aleatorio, ralentiza la carga
//pero las preguntas son mas aleatorias
i=0;
while(i<10000000)i++;

numAleatorio=numAzar(6); // da un numero entre 0 i n-1

System.out.println("Combinacion aleatoria generadada: " + numAleatorio);

switch(numAleatorio){ //hay 6 combinaciones posibles...
   case 0: primera="Resp1";
           segunda="Resp2";
           tercera="Resp3";
           numRespCorrecta="3";
           break;
   case 1: primera="Resp1";
           segunda="Resp3";
           tercera="Resp2";
           numRespCorrecta="2";
           break;
   case 2: primera="Resp2";
           segunda="Resp1";
           tercera="Resp3";
           numRespCorrecta="3";
           break;
   case 3: primera="Resp2";
           segunda="Resp3";
           tercera="Resp1";
           numRespCorrecta="2";
           break;
   case 4: primera="Resp3";
           segunda="Resp1";
           tercera="Resp2";
           numRespCorrecta="1";
           break;
   case 5: primera="Resp3";
           segunda="Resp2";
           tercera="Resp1";
           numRespCorrecta="1";
           break;

}
```

```
    //formato: categoria|pregunta|respa|respb|respc|numrespbuena
    //ejemplo de trama valida
    //"Ermua hoy|Carlos Totorika Izagirre es:|Diputado Socialista por Bizkaia|Secretario General del PSOE-EE de Bizkaia|Alcalde de Ermua|3";
        trama="";
        trama=resultado.getString("Categoria"); //categoria
        trama=trama+ConstantesServidor.SEPARADOR;
        trama=trama+ resultado.getString("Pregunta"); //pregunta
        trama=trama+ConstantesServidor.SEPARADOR;
        trama=trama+ resultado.getString(primera); // resp1
        trama=trama+ConstantesServidor.SEPARADOR;
        trama=trama+ resultado.getString(segunda); // resp2
        trama=trama+ConstantesServidor.SEPARADOR;
        trama=trama+ resultado.getString(tercera); // resp3
        trama=trama+ConstantesServidor.SEPARADOR;
        trama=trama+ numRespCorrecta; //respuesta correcta

        aAux.add(trama);
        num++;
    }
    numPreguntas=num;

        //añadimos la lista de preg/resp a la tabla
        coleccionPreguntas.add(aAux);
        con.close();
  System.out.println ("Se han cargado " + numPreguntas + " preguntas correctamente");
    }//try


    catch (ClassNotFoundException ex) {
    System.err.println("No se ha encontrado el driver para la BD.");
    System.err.println(ex);
}



    catch (Exception ex)
    {
    System.out.println("No se pudo realizar la conexion con la base de datos. "+ex.getMessage());
    System.exit(-1);
    }
  }
```

```
 ////////////////////////////////////
// funcion que devuelve un numero aleatorio entre 0 y "numero"
////////////////////////////////////
int numAzar ( int numero){
int result;
Random rnd = new Random();
result=rnd.nextInt(numero); //genera un numero aleatorio entre 0 y lo que recibe
return result;
}


}
```

Trivial.java

```
import java.net.*;
import java.io.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.border.*;


////////////////////////////////////////////////////////////////////////////
///  CLASE TRIVIAL
//APPLET PRINCIPAL, PANTALLA DE LOGIN
////////////////////////////////////////////////////////////////////////////

public class Trivial extends JApplet   {


  ////////////////////////
 ////// TIPOS DE LETRA
 ////////////////////////
 Font comic = new Font("Comic Sans MS",Font.PLAIN,11 );
 Font verdana=new Font("Verdana",Font.BOLD,14);

 ///////////////////////
 // COLORES, tocando aqui podemos cambiar facilmente el aspecto de la pantalla.
 // simplemente cambiamos los colores RGB por los que creamos apropiados
 ////////////////////////

 Color cFondoPantalla = new Color(255,0,0);
 Color cFondoPantallaGeneral= new Color(255,255,255);
 Color cFondoTitulo = new Color(173,3,11);
 Color cLetrasChat= new Color(255,255,255);
 Color cTituloBorde= new Color(243,120,126);
 Color cLineaBorde=new Color(0,0,0);
 Color cBotonSeleccionado=new Color(255,255,255);
 Color cBotonSeleccionadoRespuesta=new Color(5,255,5);

  ////////////////////
  // BORDES
  ////////////////////
```

```
Border bEspacio=  BorderFactory.createEmptyBorder(10,10,10,10);
Border bEspacioFoto=  BorderFactory.createEmptyBorder(5,5,5,5);
Border bUnEspacio=  BorderFactory.createEmptyBorder(0,5,0,5);
Border bEspacioizdoderecho=  BorderFactory.createEmptyBorder(0,10,0,10);
Border bLineaExterior=  BorderFactory.createLineBorder(cLineaBorde, 1);
Border bLineaBlanca=  BorderFactory.createLineBorder(cLineaBorde, 1);
Border bOpcionSeleccionadaPrevio=  BorderFactory.createLineBorder(cBotonSeleccionadoRespuesta, 1);
Border bOpcionSeleccionada=BorderFactory.createCompoundBorder(bOpcionSeleccionadaPrevio,
bUnEspacio);
Border bTituloLogin= BorderFactory.createTitledBorder(bLineaBlanca, "Login", TitledBorder.LEFT,
TitledBorder.ABOVE_TOP, verdana, cTituloBorde);
Border bLogin= BorderFactory.createCompoundBorder(bLineaExterior,bTituloLogin);


//////////////////////////
//variables globales
//////////////////////////

Socket s; //socket que conecta con el servidor
DataOutputStream o;
DataInputStream i;
ClienteHandler ch; // puntero a la clase clientehandler (sub-applet que contiene el juego en si)

//clase que contendra las fotografias
//es publica para que la clase clientehandler pueda acceder y mostrar la foto
public ImageIcon iconologin;

int entrar; // permitira saber si el usuario ya se ha autentificado.
int numFoto;
String sNombreFich;
String sExtension;

javax.swing.JLabel jLUsername;
javax.swing.JLabel jLPassword;
javax.swing.JLabel jLNombre;
javax.swing.JPasswordField jPasswordField1;
javax.swing.JTextField jTNombre;
javax.swing.JTextField jTUsername;
javax.swing.JLabel jLEmail;
javax.swing.JTextField jTEmail;
javax.swing.JButton JBenviar;
javax.swing.JButton JBRegistro;

JPanel jPlogin= new JPanel();
```

```
JPanel jPBordeLogin= new JPanel();
JPanel jPColumna1= new JPanel();
JPanel jPColumna2= new JPanel();
JPanel jPColumna3= new JPanel();

 int crear;
 int errores=0;  //al llegar al numero maximo de errores de login el programa se cierra
  protected Thread listener;

//////////////////////////////////////////
// Metodo que captura fotografias del servidor web.
// Es imprescindible que este en el applet principal (el que se llama desde el navegador,
// en caso contrario la JVM lanza una excepcion de seguridad al no poder determinar si la URL
// es o no la de origen.
//////////////////////////////////////////
public void pedirFotografia(String foto){
   // Capturamos una nueva fotografia.
 iconologin = new ImageIcon(getURL(getParameter("directorio_fotos")+"/"+foto));
}


////////////////////////////////////////////
/// INIT()
// Metodo que se lanza desde el navegador. Inicia el sistema.
////////////////////////////////////////////
    public void init() {

        URL host;
        String IP;
        int PUERTO;
        InetAddress ia;

      ////////////////////////
      /// APARIENCIA BORDES ETC
      ////////////////////////
       jPBordeLogin.setBorder(bLogin);
       jPBordeLogin.setBackground(cFondoTitulo);
       jPlogin.setBorder(bEspacio);
      jPlogin.setBackground(cFondoPantalla);
      jPColumna1.setBackground(cFondoPantalla);
      jPColumna2.setBackground(cFondoPantalla);
      jPColumna3.setBackground(cFondoPantalla);
```

```
try{

//URL DONDE RESIDE EL APPLET (basica, para calcular la ip)
host=this.getCodeBase();

//cojemos el host donde reside el applet y lo traducimos a su ip
//para asi conectar con el servidor de la aplicacion

ia = InetAddress.getByName(host.getHost());
IP=ia.getHostAddress();
PUERTO= Integer.parseInt(getParameter("puerto"));

//cojemos parametros que nos serviran mas adelante para visualizar las fotos
 numFoto=Integer.parseInt(getParameter("numero_de_fotos")); //da num entre 0 i num-1
 sNombreFich=getParameter("nombre_de_fichero_foto");
 sExtension=getParameter("extension_foto");

//creamos el socket para conectar con el servidor del juego
//permitira realizar la autentificacion de user y pass

s = new Socket (IP,PUERTO);
o = new DataOutputStream (new BufferedOutputStream (s.getOutputStream()));
i =  new DataInputStream  (new BufferedInputStream  (s.getInputStream()));

jLUsername = new javax.swing.JLabel();
jTUsername = new javax.swing.JTextField(12);
jLPassword = new javax.swing.JLabel("password");
jPasswordField1 = new javax.swing.JPasswordField(12);
jLNombre = new javax.swing.JLabel();
jTNombre = new javax.swing.JTextField(50);
jLEmail= new javax.swing.JLabel();
jTEmail= new javax.swing.JTextField(50);
JBenviar=new javax.swing.JButton();
JBRegistro=new javax.swing.JButton();



 ////////////////////////////
/// PANEL PRINCIPAL
// apila de arriba a abajo
////////////////////////////

//COJEMOS LA RESOLUCION ACTUAL Y SE LA ASIGNAMOS...
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
```

```
this.setSize(screenSize);

this.getContentPane().setBackground(cFondoPantallaGeneral);
this.getContentPane().setLayout(new BoxLayout(this.getContentPane(), BoxLayout.Y_AXIS));
this.getContentPane().add(Box.createRigidArea(new Dimension(0,80)));

this.getContentPane().add(jPBordeLogin);


///////////////
//zona de preguntas, superior derecha
//contenedor para poder poner el borde con el titulo
///////////////

jPBordeLogin.setLayout(new BoxLayout(jPBordeLogin, BoxLayout.Y_AXIS));
jPBordeLogin.setPreferredSize(new Dimension(410, 330));
jPBordeLogin.setMinimumSize(new Dimension(410, 330));
jPBordeLogin.setMaximumSize(new Dimension(410, 330));
jPBordeLogin.add(jPlogin);


jPlogin.setLayout(new BoxLayout(jPlogin, BoxLayout.X_AXIS));
jPlogin.setPreferredSize(new Dimension(400, 300));
jPlogin.setMinimumSize(new Dimension(400, 300));
jPlogin.setMaximumSize(new Dimension(400, 300));
jPlogin.add(jPColumna1);
jPlogin.add(Box.createRigidArea(new Dimension(25,0)));
jPlogin.add(jPColumna2);
jPlogin.add(Box.createRigidArea(new Dimension(25,0)));



 ///////////////
// zona de etiquetas (username, password, etc).
///////////////
//contenedor de arriba a abajo
 jPColumna1.setLayout(new BoxLayout(jPColumna1, BoxLayout.Y_AXIS));
 jPColumna1.setPreferredSize(new Dimension(100, 300));
 jPColumna1.setMinimumSize(new Dimension(100, 300));
 jPColumna1.setMaximumSize(new Dimension(100, 300));

 jPColumna1.add(Box.createRigidArea(new Dimension(0,50)));
 jPColumna1.add(jLUsername);
 jPColumna1.add(Box.createRigidArea(new Dimension(0,15)));
```

```
 jPColumna1.add(jLPassword);
 jPColumna1.add(Box.createRigidArea(new Dimension(0,15)));
 jPColumna1.add(jLNombre);
 jPColumna1.add(Box.createRigidArea(new Dimension(0,17)));
 jPColumna1.add(jLEmail);
 jPColumna1.add(Box.createRigidArea(new Dimension(0,38)));
 jPColumna1.add(JBenviar);
 jPColumna1.add(Box.createRigidArea(new Dimension(0,10)));
 jPColumna1.add(JBRegistro);
 //jPColumna1.add(Box.createRigidArea(new Dimension(0,10)));

 ////////////////
// zona de campos de texto (username, password, etc).
///////////////
//contenedor de arriba a abajo
 jPColumna2.setLayout(new BoxLayout(jPColumna2, BoxLayout.Y_AXIS));
 jPColumna2.setPreferredSize(new Dimension(150, 300));
 jPColumna2.setMinimumSize(new Dimension(150, 300));
 jPColumna2.setMaximumSize(new Dimension(150, 300));

 jPColumna2.add(Box.createRigidArea(new Dimension(0,45)));
 jPColumna2.add(jTUsername);
 jPColumna2.add(Box.createRigidArea(new Dimension(0,12)));
 jPColumna2.add(jPasswordField1);
 jPColumna2.add(Box.createRigidArea(new Dimension(0,13)));
 jPColumna2.add(jTNombre);
 jPColumna2.add(Box.createRigidArea(new Dimension(0,12)));
 jPColumna2.add(jTEmail);
 jPColumna2.add(Box.createRigidArea(new Dimension(0,125)));

//////////////////////////////////////
/// ASPECTO BOTONES
//////////////////////////////////////

jLUsername.setBackground(cFondoPantalla);
jLUsername.setForeground(cLetrasChat);
jLUsername.setFont(comic);

jTUsername.setBackground(cFondoPantalla);
jTUsername.setForeground(cLetrasChat);
jTUsername.setFont(comic);

jLPassword.setBackground(cFondoPantalla);
jLPassword.setForeground(cLetrasChat);
```

```
jLPassword.setFont(comic);

jPasswordField1.setBackground(cFondoPantalla);
jPasswordField1.setForeground(cLetrasChat);
jPasswordField1.setFont(comic);

jLNombre.setBackground(cFondoPantalla);
jLNombre.setForeground(cLetrasChat);
jLNombre.setFont(comic);

jTNombre.setBackground(cFondoPantalla);
jTNombre.setForeground(cLetrasChat);
jTNombre.setFont(comic);

jLEmail.setBackground(cFondoPantalla);
jLEmail.setForeground(cLetrasChat);
jLEmail.setFont(comic);

jTEmail.setBackground(cFondoPantalla);
jTEmail.setForeground(cLetrasChat);
jTEmail.setFont(comic);

JBenviar.setFont(comic);
JBRegistro.setFont(comic);

jLUsername.setText("Nick");
jTUsername.setText("");
jLPassword.setText("Password");
jLNombre.setText("Nombre completo");
jLEmail.setText("E-mail");
JBenviar.setText(" Entrar ");
JBRegistro.setText("Registro");

this.setVisible(true);


////////////////////////////////////
//EVENTO QUE OCURRE AL PULSAR "ENVIAR"
//llamamos a la clase cliente que abrira un socket y comenzara el juego
//recibiendo ademas info de user y pass, etc.
////////////////////////////////////

 JBenviar.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
```

```
        try{
       JBenviar.setEnabled(false);
       //comprobar que no ha dejado campos en blanco...
       if ( (jTUsername.getText().equals("")) || (jPasswordField1.getPassword().equals("")) ){
        JOptionPane.showMessageDialog(null, "Los campos Nick y password son necesarios para poder jugar.","Trivial OnLine", JOptionPane.ERROR_MESSAGE);
        JBenviar.setEnabled(true);
       }
       else{

       CrearCliente(jTUsername.getText(),jTNombre.getText());
        }
       }
        catch(Exception ex){

        }
     }
   });


   ////////////////////////
   //evento que ocurre al pulsar el boton de REGISTRO...
   ////////////////////////
   JBRegistro.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
        try{
          JBRegistro.setEnabled(false);


       if ( (jTUsername.getText().equals("")) || (jPasswordField1.getPassword().equals("")) || (jTNombre.getText().equals("")) || (jTEmail.getText().equals(""))){
       JOptionPane.showMessageDialog(null, "Los campos Nick, password, nombre y e-mail son necesarios para poder registrarse.","Trivial OnLine", JOptionPane.ERROR_MESSAGE);
        JBRegistro.setEnabled(true);
       }
       else{

         //miramos que los campos cumplan la longitud minima
          String nick = jTUsername.getText();
          char[] password = jPasswordField1.getPassword();

          if ( (nick.length()<ConstantesCliente.MINCARACTERES) || (password.length<ConstantesCliente.MINCARACTERES)){
```

```
        JOptionPane.showMessageDialog(null, "La longitud minima de los campos Nick y password es de " +
ConstantesCliente.MINCARACTERES + " caracteres","Trivial OnLine", JOptionPane.ERROR_MESSAGE);
        }
        else{
        //enviamos info para registrar el usuario.

        int                                                    result=registrarUsuario(jTUsername.getText(),new
String(jPasswordField1.getPassword()),jTNombre.getText(), jTEmail.getText());
        switch(result){
           case 0:  //el usuario ya existe
                  JOptionPane.showMessageDialog(null, "El nick que ha escogido ya esta siendo utilizado, por
favor registrese con otro nick.","Trivial OnLine", JOptionPane.ERROR_MESSAGE);
                  jTUsername.setText("");
                  jPasswordField1.setText("");
                  break;
           case 1:  //registro correcto ya puede hacer login
                  JOptionPane.showMessageDialog(null, "Registro correcto. Ya puede entrar al juego.","Trivial
OnLine",JOptionPane.DEFAULT_OPTION);

                 //borramos los campos
                 jTNombre.setText("");
                 jTEmail.setText("");

                  break;

        } //switch
       }//else
        JBRegistro.setEnabled(true);
       }

       }
       catch(Exception ex){

       }
    }
  });

     }
    catch (Exception e) {

    System.out.println("ERROR: No es posible conectar con el servidor del juego.");
    System.exit(-1);
  }
```

```
    }

  ////////////////////////////////////////////////////
  /// ENVIAMOS UNA TRAMA DE AVISO CONFORME HEMOS SALIDO
  /////////////////////////////////////////////////
  public void notificarSalidaSinNick(){
     try{
    String resp;
    resp=ConstantesCliente.ID_TRAMA_SALIDASINNICK;
    o.writeUTF (resp);
    o.flush ();
    }
     catch (IOException ex) {
          ex.printStackTrace();

          }
}


   ////////////////////////////////////////
   ////// crea la clase CLIENTE HANDLER que contiene toda la logica de la aplicacion
   ///////////////////////////////////////
   public void CrearCliente(String user,String ip) throws IOException{
      try  {

      entrar=Autentificacion(jTUsername.getText(), new String(jPasswordField1.getPassword()));


  if (entrar==1){    //clave correcta, empieza el juego

     //creamos el applet del juego, pasandole info de conexion con el server y el username

    ch= new ClienteHandler();

    //hacemos que el contenedor swing principal sea el de clientehandler

    this.getContentPane().removeAll();
    this.getContentPane().add(ch);

    //inicalizamos el applet del juego
    ch.init(s.getInputStream()                    ,                    s.getOutputStream
(),user,numFoto,sNombreFich,sExtension,getParameter("directorio_fotos"),this);
    ch.start();
    this.validate();
```

```
        } //if

  else { //clave incorrecta, damos mensaje de error...
  if (entrar==2){
     JOptionPane.showMessageDialog(null, "Solo se puede jugar 1 partida por usuario","Trivial OnLine",
JOptionPane.ERROR_MESSAGE);
     }
     else{
     JOptionPane.showMessageDialog(null, "Usuario inexistente y/o contraseña incorrecta.","Trivial OnLine",
JOptionPane.ERROR_MESSAGE);
     }//else interior
     JBenviar.setEnabled(true);
     jTUsername.setText("");
     jPasswordField1.setText("");
     errores++;

      }//else

      }//try

   catch (Exception e) {
    e.printStackTrace();
  }
  }

 ////////////////////////////////////////////
 // METODO QUE SE EJECUTA CUANDO EL USUARIO CIERRA EL NAVEGADOR
 // aprovecharemos para enviar trama de desconexion al servidor.
 // (si estamos en clientehandler salta tambien este mismo metodo)
 ////////////////////////////////////////////

  public void stop(){
      if (entrar==1){
         //si ya nos hemos autentificado comunicamos el nick al servidor para que lo elimine
         //de la lista de usuarios
           ch.notificarSalida();
      }
      else{
     notificarSalidaSinNick();
      }
 }
```

```
    ////////////////////
    //enviar una trama para poder registrar un nuevo jugador
    ////////////////////
    public int registrarUsuario(String user,String pass,String nombre,String email){
    try{

     String resp;

     //enviamos trama
     resp=ConstantesCliente.ID_TRAMA_REGISTRO + getParameter("directorio_fotos") +
ConstantesCliente.SEPARADOR + user + ConstantesCliente.SEPARADOR + pass+
ConstantesCliente.SEPARADOR + nombre +ConstantesCliente.SEPARADOR +email;
     //montamos la trama
     o.writeUTF (resp);
     o.flush ();

     //esperamos la respuesta, hasta que no recibamos la trama de respuesta nos esperamos
     //hay que esperar puesto que se reciben mas tramas, de las preguntas puntuaciones y demas
     //que hay que descartar...
     while (true){
        String line = i.readUTF ();

        if (line.substring(0,4).equals(ConstantesCliente.ID_TRAMA_REGISTRO_CORRECTO)){

          if (line.charAt(4)=='1'){  //OK

             return 1;
          }
          else if (line.charAt(4)=='0'){
             return 0;
          }
        }//if trama


     }//while
    }//try
     catch (IOException ex) {
          ex.printStackTrace();
             return(-1);
          }
    }
```

```
//////////////////////
//enviamos trama con tematica, user y pass al servidor, esperamos la respuesta del servidor...
//////////////////////
int Autentificacion (String user, String pass){

try{

String resp;

//enviamos trama
resp=ConstantesCliente.ID_TRAMA_AUTENTIFICACION + getParameter("directorio_fotos") + ConstantesCliente.SEPARADOR + user + ConstantesCliente.SEPARADOR + pass; //montamos la trama
o.writeUTF (resp);
o.flush ();

//esperamos la respuesta, hasta que no recibamos la trama de respuesta nos esperamos
//hay que esperar puesto que se reciben mas tramas, de las preguntas puntuaciones y demas
//que hay que descartar...
while (true){
   String line = i.readUTF ();

   if (line.substring(0,4).equals(ConstantesCliente.ID_TRAMA_PERMISOPARAJUGAR)){


     if (line.charAt(4)=='1'){  //OK

        return 1;
     }
     else if (line.charAt(4)=='2'){  //YA ESTA JUGANDO, NO PUEDE JUGAR 2 VECES
        return 2;
     }
     else {   //CONTRASEÑA INCORRECTA

        return 0;
     }
   }

}//while
}//try
catch (IOException ex) {
      ex.printStackTrace();
        return(-1);
```

```
        }
    }


    ////////////////////////////////////////
    // GET URL
    // Recibe el nombre de fichero de la fotografia a cargar y devuelve la URL absoluta
    // para poder realizar la carga en un objeto imagen
    ////////////////////////////////////////

    protected URL getURL(String filename) {
        URL codeBase = this.getCodeBase();
        URL url = null;

        try {
            url = new URL(codeBase, filename);
        } catch (java.net.MalformedURLException e) {
            System.out.println("No se ha podido cojer la imagen, URL mal construida");
            return null;
        }
        return url;
    }

}
```

Conclusión

El resultado final del trabajo ha sido realmente enriquecedor ya que ha permitido ampliar los conocimientos sobre la tecnología Java y todo lo relacionado con acceso a base de datos y comunicaciones en red, y se han adquirido suficientes conocimientos como para desarrollar una aplicación de complejidad media-alta. También ha servido como repaso y puesta en práctica de numerosos conceptos alcanzados en diversas asignaturas a lo largo de la carrera, especialmente las asignaturas de programación, bases de datos, ingeniería del software y redes.

Se puede decir que se han alcanzado todos los objetivos propuestos inicialmente, tanto por el desarrollo de la aplicación como por los conocimientos adquiridos. Por otro lado, la propia redacción de la memoria es un ejercicio recomendable para alcanzar una base sólida en redacción de documentación técnica.

Bibliografía

Fundamentos de programación. Materiales de la asignatura. Universitat Oberta de Catalunya.

Seguridad de redes. Materiales de la asignatura. Universitat Oberta de Catalunya.

Bases de datos. Materiales de la asignatura. Universitat Oberta de Catalunya.

Sistemas operativos. Materiales de la asignatura. Universitat Oberta de Catalunya.

Ingeniería del software. Materiales de la asignatura. Universitat Oberta de Catalunya.

Bibliografía complementaria

Eckel, Bruce (2006) "Piensa en Java"

Schildt, Herbert (2005) "Java 2 v5.0"

Holzner, Steven (2000) "La biblia de Java"

Montero, Roberto (2011) "Java 7"