Programación en Entornos Distribuidos

Práctica 2 – Chat Básico

“PEDChat”

Fausto Escrigas Pérez

Madrid, 1/06/2012

Índice

Introducción

Inicialmente se pretende desarrollar un chat básico. Se ha cumplido con la funcionalidad mínima requerida por el profesor y su vez se han implementado algunas de las extensiones planteadas en el enunciado.

A continuación se detallará brevemente el contenido de cada uno de los apartados de esta memoria:

En el primer apartado “Diseño: UML”, se mostrará los diagramas UML que se han seguido para la elaboración de ambos programas (cliente y servidor) y se intentará explicar el por qué de los mismos.

En el segundo apartado “Diseño BD: Diagrama E/R” se incluye el diagrama entidad relación diseñado para la base de datos que emplea la aplicación, y los scripts necesarios para crearla.

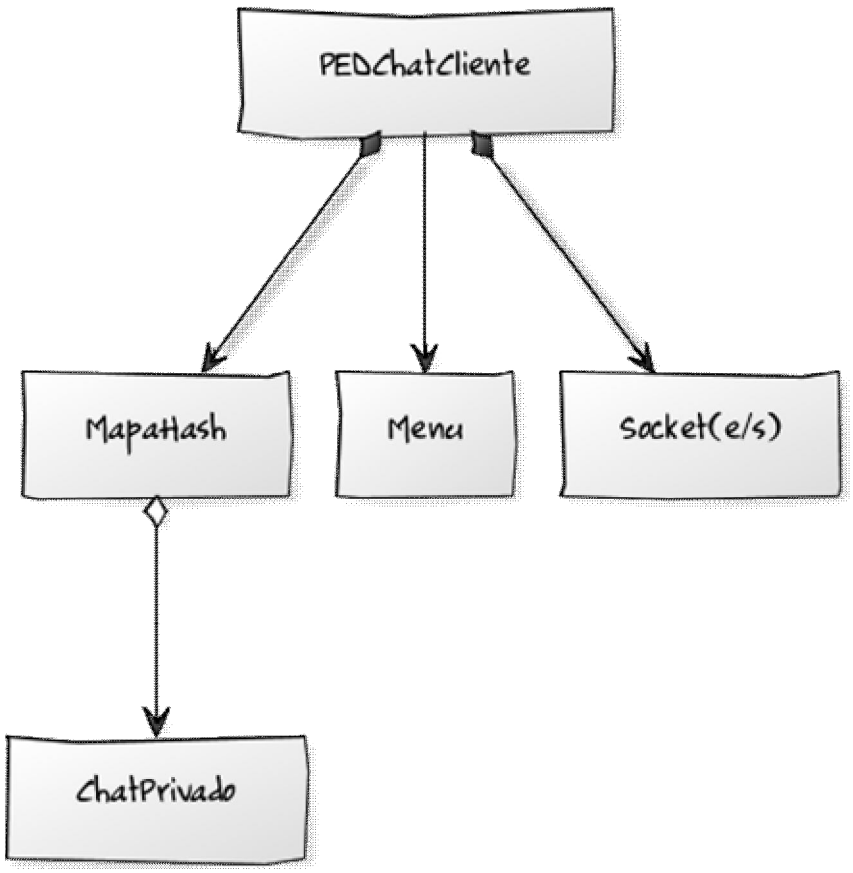
En el tercer apartado “Protocolo Empleado” se especificará el protocolo que se ha diseñado para que cliente y servidor puedan mantener un diálogo y ejecutar las diferentes acciones ordenadas por el usuario.

En el cuarto apartado “Extensiones Implementadas” se enumerarán y detallarán las diferentes extensiones propuestas en el enunciado, que finalmente han sido incluidas en el proyecto.

Finalmente se incluirá el código fuente tanto del programa cliente como del servidor.

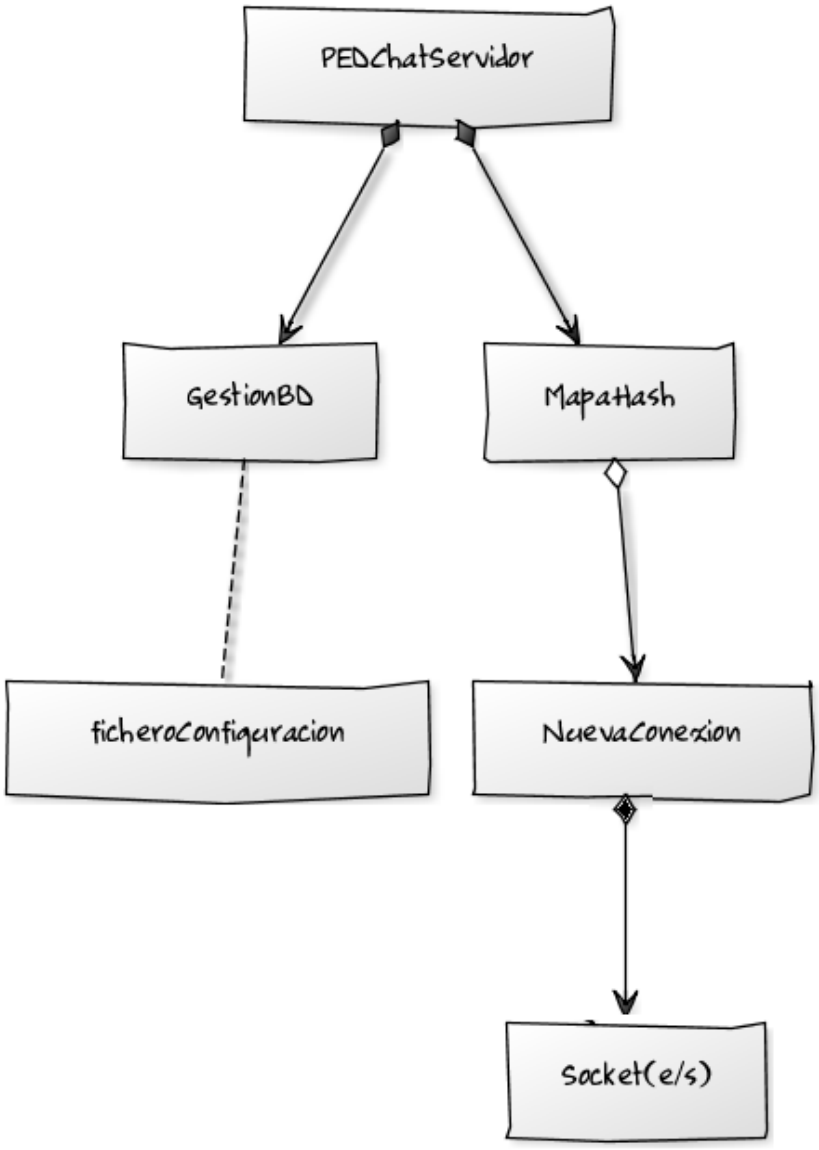
Diseño: Diagramas UML de Clases

PEDChat_Cliente



PEDChat_Cliente es la clase principal del lado cliente de la aplicación, es la encargada de gestionar las conversaciones privadas del cliente empleando para ello un **MapaHash<idConversación, ChatPrivado>** que relaciona los identificadores de cada conversación con el **ChatPrivado** en el que tiene lugar. El intercambio de comandos y/o objetos tiene lugar a través de los Streams de Entrada/Salida que ofrece la conexión del **Socket**, por último podemos decir que esta clase intercambia información con la clase **Menu** (interfaz gráfica), a tavés de métodos que le permiten modificarla en función de las entradas recibidas.

PEDChat_Servidor



La clase **PEDChat_Servidor** es la clase principal del lado Servidor de la aplicación, y el pilar fundamental de la aplicación al completo. Gestiona las diferentes conexiones de los clientes al chat (Threads, **NuevasConexiones**) con el objetivo comunicarlos entre sí, empleando para ello una estructura de **MapasHash** que le permiten asociar cada usuario con el thread correspondiente y los diferentes chats privados en los que participa. Además es la encargada de interactuar con la Clase **GestionBaseDatos** registrando usuarios, mensajes, etc.

Diseño BD: Diagrama Entidad - Relación

De manera conceptual la base de datos con la que trabaja la aplicación (lado Servidor, Clase GestionBaseDatos) obedece al siguiente diagrama Entidad – Relación:

Diagrama Lógico

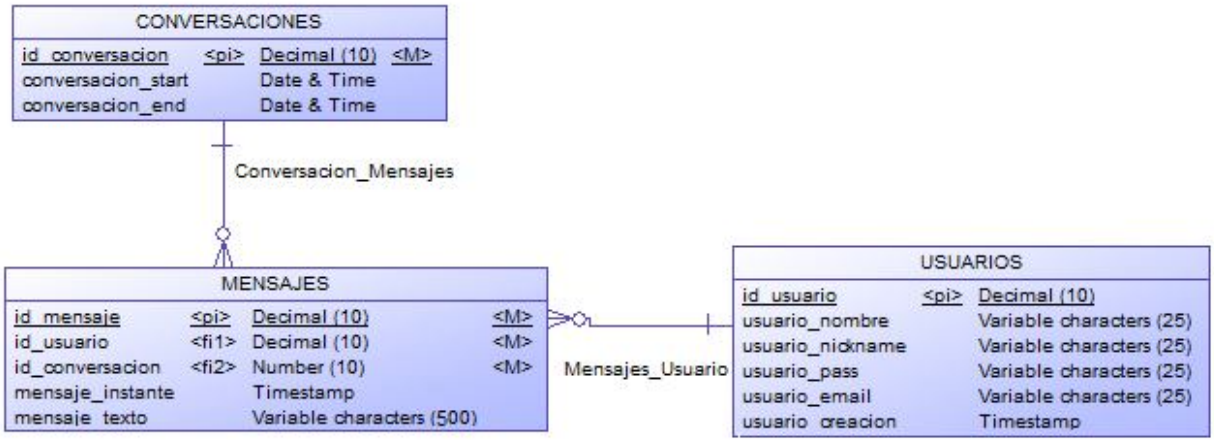
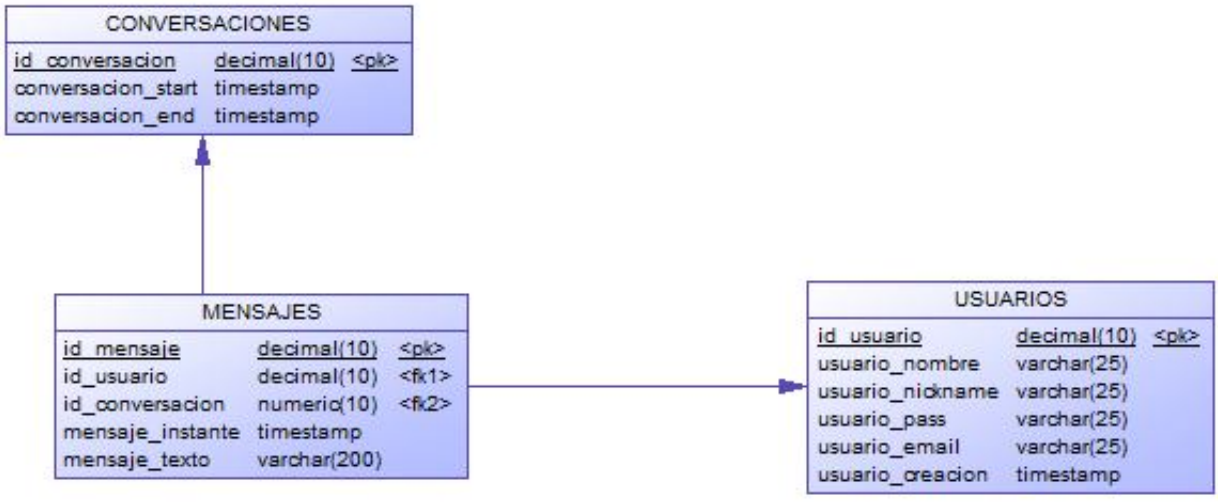


Diagrama Físico



Diseño BD: Scripts

Debemos ejecutar el siguiente Script SQL para generar la base de datos que emplea la aplicación.

```
create table CONVERSACIONES
(
  ID_CONVERSACION      varchar(40) NOT NULL,
  CONVERSACION_START   TIMESTAMP,
  CONVERSACION_END     TIMESTAMP,
  PRIMARY KEY (ID_CONVERSACION)
);

create table USUARIOS
(
  ID_USUARIO          INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1,
INCREMENT BY 1),
  USUARIO_NOMBRE       VARCHAR(25),
  USUARIO_NICKNAME     VARCHAR(25),
  USUARIO_PASS         VARCHAR(25),
  USUARIO_EMAIL        VARCHAR(25),
  USUARIO_CREACION     TIMESTAMP,
  PRIMARY KEY (ID_USUARIO)
);


create table MENSAJES
(
  ID_MENSAJE          INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1,
INCREMENT BY 1),
  ID_USUARIO          INTEGER NOT NULL,
  ID_CONVERSACION      varchar(40) NOT NULL,
  MENSAJE_INSTANTE     TIMESTAMP,
  MENSAJE_TEXTO        VARCHAR(200),
  PRIMARY KEY (ID_MENSAJE)
);

ALTER TABLE MENSAJES
  ADD FOREIGN KEY(ID_CONVERSACION)
    REFERENCES CONVERSACIONES (ID_CONVERSACION);
```

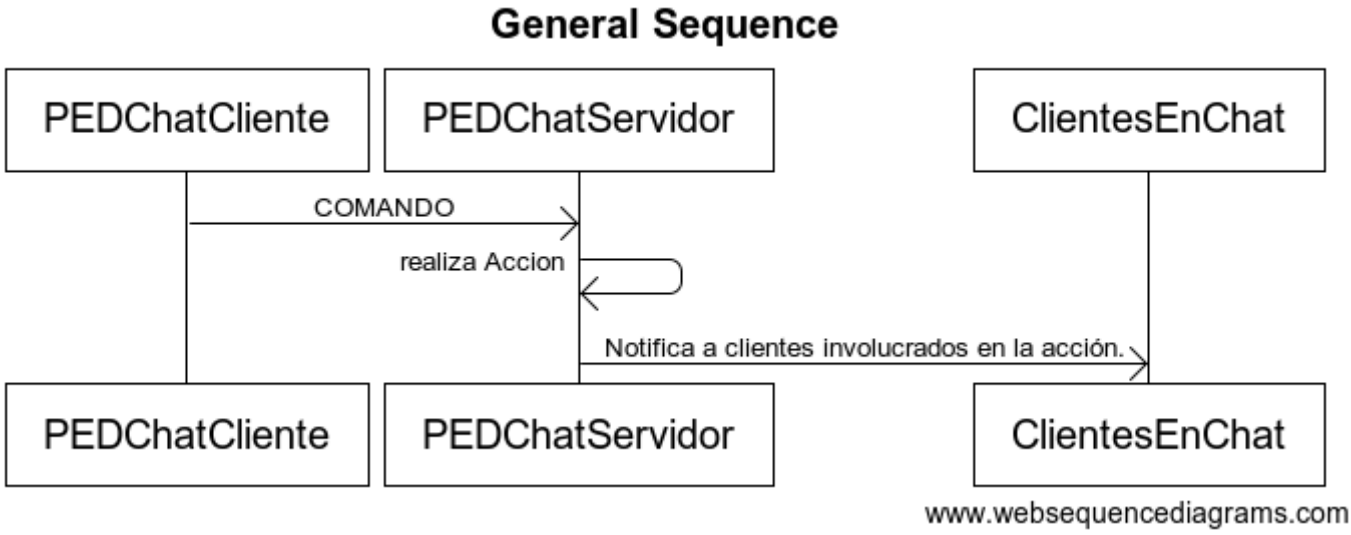
```
ALTER TABLE MENSAJES
  ADD FOREIGN KEY(ID_USUARIO)
    REFERENCES USUARIOS (ID_USUARIO);

/* Iniciamos el chat publico, CONVERSACION_END nos mostrará la última vez que el chat se quedo
vacío. */
INSERT INTO CONVERSACIONES
(ID_CONVERSACION,CONVERSACION_START,CONVERSACION_END) VALUES
('0000',CURRENT_TIMESTAMP,CURRENT_TIMESTAMP);
```

Protocolo Empleado

A continuación y través de diagramas de secuencia se especificará el protocolo empleado por cliente y servidor para llevar a cabo su funcionamiento.

En líneas generales podemos decir que para cada uno de los mensajes se sigue la secuencia:



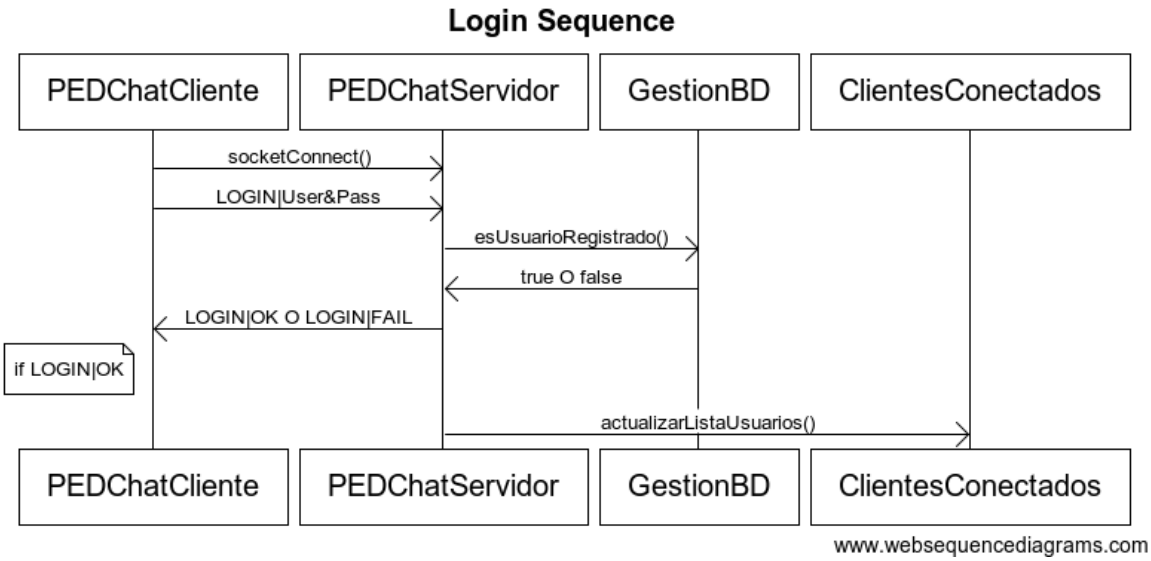
El cliente conectado al Servidor, desea realizar alguna de las acciones disponibles, como enviar un mensaje o solicitar a una persona del Chat una conversación privada. Para ello notifica su intención al Servidor enviándole el comando apropiado, el servidor lo recibe, realiza la acción solicitada y a continuación la notifica al resto de los usuarios involucrados en la acción, reenviando otro comando y/o objeto.

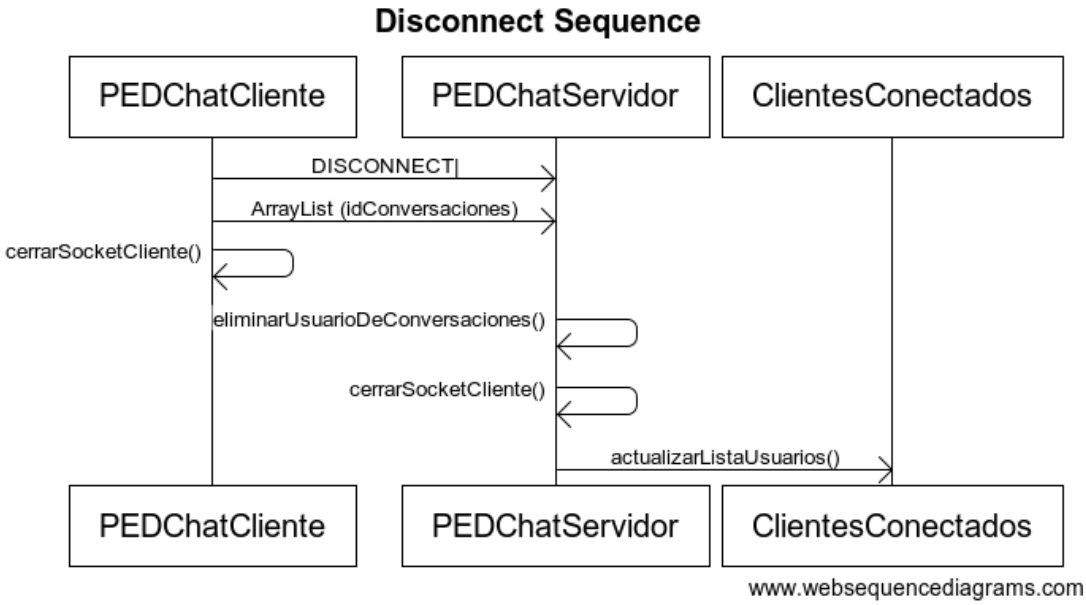
El formato empleado por los comandos tiene la siguiente estructura:

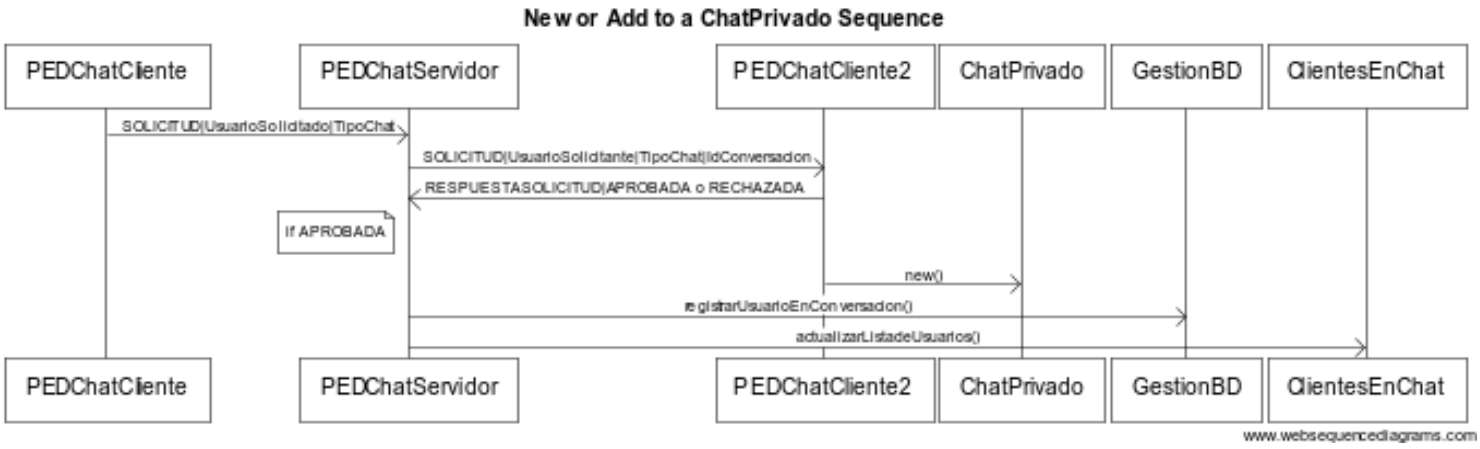
`COMANDO|CONTENIDO`

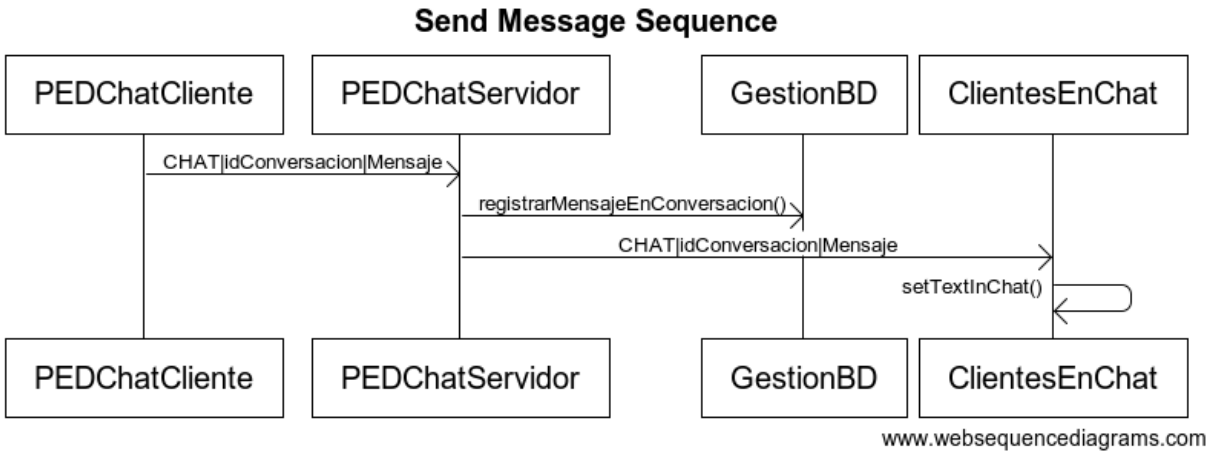
En ocasiones el CONTENIDO del comando incluye una serie de parámetros necesarios para ejecutar la instrucción solicitada.

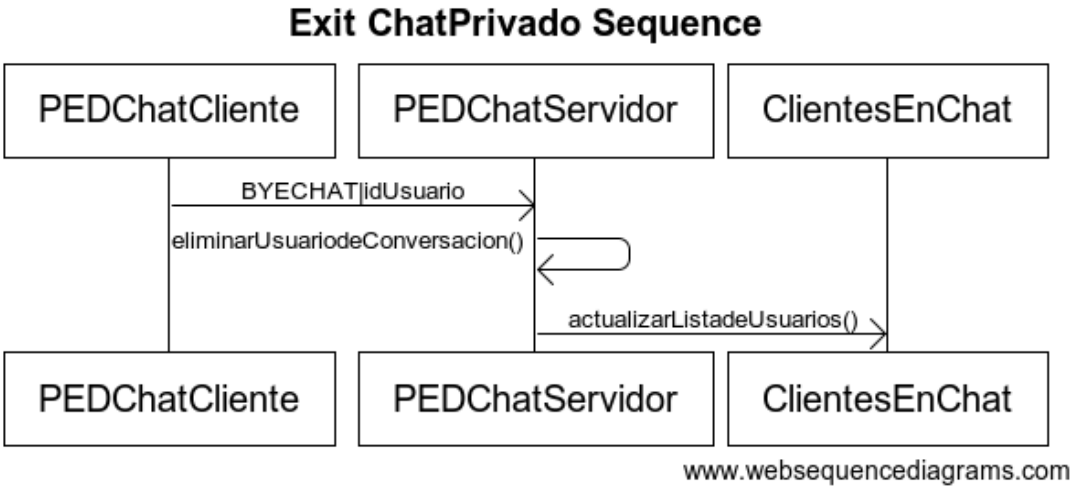
A continuación se presentan de manera esquemática los diagramas de secuencia que representan las principales acciones que puede realizar un usuario de la aplicación:






New or Add to a ChatPrivado Sequence
PEDChatCliente
PEDChatServidor
PEDChatCliente2
ChatPrivado
GestionBD
ClientesEnChat
SOLICITUD|UsuarioSolicitado|TipoChat
SOLICITUD|UsuarioSolicitante|TipoChat|idConversacion
RESPUESTASOLICITUD|APROBADA o RECHAZADA
If APROBADA
new()
registrarUsuarioEnConversacion()
actualizarListadeUsuarios()
www.websequencediagrams.com


Send Message Sequence
PEDChatCliente
PEDChatServidor
GestionBD
ClientesEnChat
CHAT|idConversacion|Mensaje
registrarMensajeEnConversacion()
CHAT|idConversacion|Mensaje
setTextInChat()
www.websequencediagrams.com


Exit ChatPrivado Sequence
PEDChatCliente
PEDChatServidor
ClientesEnChat
BYECHAT|idUsuario
eliminarUsuariodeConversacion()
actualizarListadeUsuarios()
www.websequencediagrams.com

Extensiones Implementadas

De las posibles extensiones planteadas por el profesor para mejorar la práctica han sido implementadas las siguientes:

- *Permitir que de modo dinámico nuevos participantes se unan o salgan de una conversación multi-chat (hasta 0.5 puntos extra).*

  La aplicación emplea el mismo comando para iniciar un chat privado 'simple' (de dos personas) que para agregar un nuevo usuario (chat privado múltiple) a un chat privado ya existente. Desde la ventana del chat privado es posible incluir nuevos usuarios a la conversación y siempre y cuando ellos acepten la invitación. Igual que sucede con el chat general, si un usuario abandona la conversación privada, esta se actualiza dinámicamente, siguiendo la conversación con los participantes que continúan en ella.

- *Desarrollar dos posibles interfases para el cliente: una basada en un sistema de ventanas, y otra basada en una consola (hasta 2 puntos extra, a depender de la calidad de ambas interfaces).*

  Sólo se ha desarrollado una interfaz gráfica en el lado del cliente.

- *Usar un sistema de control de versiones desde el principio del desarrollo de la práctica (hasta 1 punto extra).*

  Se han empleado los siguientes repositorios:

  PEDChat_Cliente
  https://bitbucket.org/Fausto_Escrigas/pedchat_cliente

  PEDChat_Servidor
  https://bitbucket.org/Fausto_Escrigas/pedchat_servidor

  Ambos están abiertos al público general y no son necesarios ningún tipo de permisos para acceder a su contenido.

- *Implementar alguna solución que permita una comunicación encriptada entre el cliente y el servidor (hasta 2 puntos extra para el primero que lo implemente; hasta 1 punto extra para los demás).*

Se ha empleado SSL Server Sockets para garantizar una comunicación más segura. Se ha empleado el mismo almacén de claves que en la práctica anterior *Fftp, en este caso este se encuentra en la siguiente ruta:

En el caso del proyecto PEDChat_Cliente:

```
PEDChat_Cliente/AlmacenClavesPED
```

En el caso del proyecto PEDChat_Servidor:

```
PEDChat_Servidor/AlmacenClavesPED
```

El Almacén de Claves fue generado empleando la herramienta *keytool:*

```
keytool  -genkey -keystore AlmacenClavesPED  -keyalg RSA  -alias
claveDeFftp

Escriba la contraseña del almacén de claves: 123456
¿Cuáles son su nombre y su apellido?
 [Unknown]:  Fausto Escrigas
¿Cuál es el nombre de su unidad de organización?
 [Unknown]:  Fftp
¿Cuál es el nombre de su organización?
 [Unknown]:  EPS USP CEU
¿Cuál es el nombre de su ciudad o localidad?
  [Unknown]:  Boadilla del Monte
 ¿Cuál es el nombre de su estado o provincia?
  [Unknown]:  Madrid
 ¿Cuál es el código de país de dos letras de la unidad?
  [Unknown]:  ES
 ¿Es correcto CN=Fausto Escrigas, OU=Fftp, O=EPS USP CEU, L=Boadilla
del Monte, ST=Madrid, C=ES?
  [no]:  si
```

Para ejecutar ambas partes de la aplicación será necesario indicar dónde se encuentra ubicados dichos almacenes para que el protocolo SSL pueda funcionar correctamente. Si arrancamos desde consola:

PEDChat_Cliente

```
java -Djavax.net.ssl.trustStore=miAlmacenDeClaves -
Djavax.net.ssl.trustStorePassword=123456 -jar PEDChat_Cliente.jar
```

PEDChat_Servidor

```
java -Djavax.net.ssl.keyStore=miAlmacenDeClaves -
Djavax.net.ssl.keyStorePassword=123456 -jar PEDChat_Servidor.jar
```

Situando el .jar y el AlmacenDeClavesPED en un mismo directorio.

Si el proyecto es arrancado desde NetBeans, la configuración ya ha sido cambiada.