

UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA



Sistemas para la Colaboración

- PLANCODE -
PLANIFICADOR COLABORATIVO PARA EL DISEÑO DE ESTRATEGIAS Y ACCIONES DE
EMERGENCIA

Juan Andrada Romero
Jose Domingo López López

10 de junio de 2010

Índice

1. Introducción	1
2. ¿Qué es PlanCoDE?	1
2.1. Diagrama de casos de uso	2
3. Clasificación	3
3.1. Dimensiones espacio-temporal	3
3.2. Dominio de aplicación	3
3.3. Modelo 3C	3
4. Descripción de los requisitos funcionales	5
4.1. Participación	5
4.2. Interacción	5
4.3. Coordinación	6
4.4. Distribución y soporte	6
4.5. Notificación de eventos	6
4.6. Visualización	6
5. Decisiones de diseño	8
5.1. Lenguaje de programación	8
5.2. Diseño multicapa	8
5.3. Desarrollo del sistema	12
5.3.1. Arquitectura de PlanCode	12
5.3.2. Canales implementados	12
5.3.3. Prototipo de la interfaz gráfica de usuario	12
5.3.4. Gestión de trazos	13
6. Manual de usuario	15
6.1. Requisitos del sistema	15
6.1.1. Entorno de Java	15
6.2. Guía de uso	16
7. Informe de división de trabajo	22
Referencias	23

1. Introducción

PlanCode es un sistema *groupware* cuyo objetivo principal es la elaboración de planes antes situaciones de emergencia de forma colaborativa. De este modo, este documento aborda el sistema *groupware* que ha sido desarrollado. En la sección 2 se detalla la especificación de requisitos de dicho sistema; en la sección 3 se realiza una clasificación de este sistema *groupware*; en la sección 4 se abordan los requisitos funcionales de un sistema *groupware* que son satisfechos por esta herramienta; en la sección 5 se plantea el diseño del sistema y, para terminar, se muestra en la sección 6 un pequeño manual de usuario de PlanCode.

2. ¿Qué es PlanCoDE?

PlanCoDE es el acrónimo de *Planificador Colaborativo para el Diseño de Estrategias y Acciones de Emergencia*. Es, por tanto, una herramienta que permite la colaboración de diferentes especialistas y cuerpos de seguridad y emergencia (policías, bomberos, etc.) para diseñar y ejecutar un plan de actuación y poder atender una situación de emergencia.

PlanCoDE ofrece las siguientes funcionalidades a sus usuarios:

- **Chat:** los diferentes miembros de grupos especialistas en situaciones de emergencia podrán utilizar el chat que la aplicación proporciona para comunicarse en tiempo real e intercambiar opiniones acerca del plan a realizar.
- **Visualización de mapas:** la aplicación mostrará una imagen del mapa de la zona donde se ha producido la emergencia. Dichos mapas se recuperarán a partir de imágenes ya almacenadas en disco.
- **Edición de mapas:** los usuarios del sistema podrán realizar trazos libres sobre la imagen del mapa que se ha cargado. De este modo, se pueden realizar rutas sobre el propio mapa, del mismo modo que se podrán eliminar.

Además, PlanCoDE debe tener como requisitos adicionales la **movilidad** y la **tolerancia a fallos** debido a las situaciones críticas en las que dicha herramienta será utilizada. Esto implica disponer de una arquitectura lo más descentralizada posible. Por ello, ésta no será la típica arquitectura cliente-servidor, ya que si en algún momento el servidor no está disponible por la razón que sea, los usuarios no podrían iniciar el sistema para diseñar las estrategias pertinentes. Por tanto, el primer usuario que inicie una sesión se desempeñará la función de servidor y cliente al mismo tiempo, permitiendo que resto de usuarios puedan conectarse a él. De modo que cuando se ejecuta la aplicación, ésta debe permitir al usuario crear una nueva sesión o unirse a una sesión existente. En el segundo caso, será necesario introducir la dirección IP y el puerto al que debe conectarse la aplicación.

Con esta arquitectura, se satisfacen los requisitos críticos de movilidad y tolerancia a fallos, pues cualquier cliente puede conectarse en cualquier momento y lugar y siempre va a existir algún servidor disponible, ya que éste será uno de los propios clientes de la aplicación.

2.1. Diagrama de casos de uso

El diagrama de casos de uso de la aplicación se muestra en la Figura 2.1. Dicho diagrama representa los requisitos de la aplicación que se han comentado en el punto anterior, junto con otras funcionalidades que son necesarias para el correcto funcionamiento de la aplicación colaborativa.

Así, un usuario debe iniciar una sesión para poder utilizar la aplicación con el resto de usuarios y cierra la sesión cuando deja de utilizarla. Del mismo modo, la propia aplicación debe actualizar su estado y enviar a todos los usuarios conectados esta información, es decir, los mensajes que se envían al chat, la imagen del mapa que se desea mostrar y los dibujos que se realizan sobre éste.

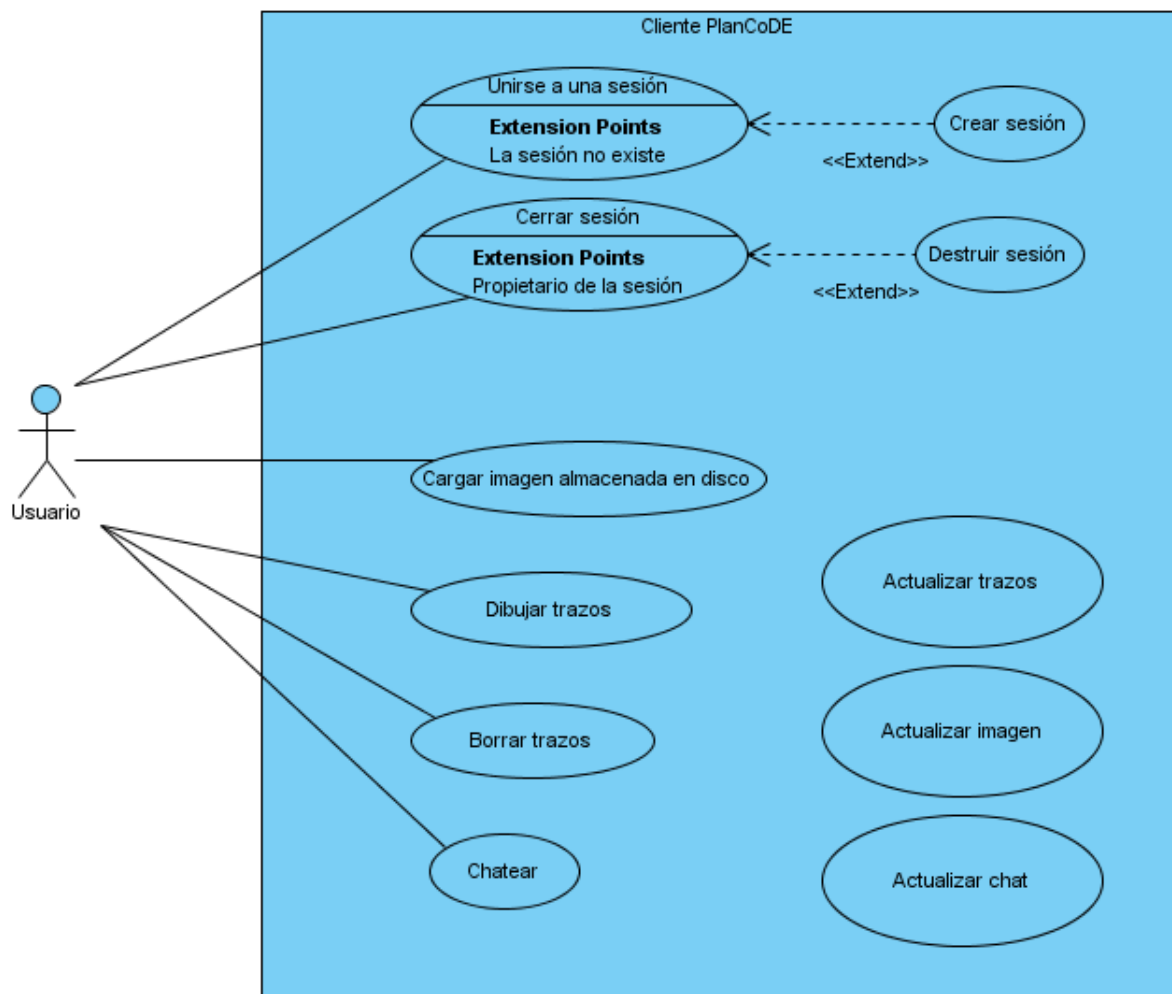


Figura 2.1: Diagrama de casos de uso de la aplicación

3. Clasificación

En los siguientes apartados, se realizará una clasificación de la herramienta PlanCode, según los diferentes autores que aparecen en [6].

3.1. Dimensiones espacio-temporal

Atendiendo a la clasificación en tiempo y espacio propuesta por **Johansen**, PlanCode puede clasificarse en la categoría *diferente lugar, mismo tiempo*, es decir, es una herramienta de interacción síncrona distribuida (ver Figura 3.1). Esto es así porque los diferentes usuarios deben estar conectados en la misma sesión y al mismo tiempo para poder recibir los mensajes del chat, la imagen del mapa cargada por alguno de ellos y los trazos que se van realizando.

	Mismo tiempo	Diferente tiempo
Mismo lugar	Interacción cara a cara	Interacción asíncrona
Diferente lugar	Interacción distribuida síncrona	Interacción distribuida asíncrona

Figura 3.1: Taxonomía en tiempo y espacio (Johansen; 1991)

Por otra parte, atendiendo a la clasificación espacio-temporal propuesta por **Grudin**, la herramienta se puede clasificar también en la categoría *diferente lugar, mismo tiempo*, por las mismas razones que en el caso anterior.

3.2. Dominio de aplicación

Siguiendo la clasificación del nivel de aplicaciones de **Ellis**, PlanCode cumple con las características de los siguientes sistemas:

- A. **Sistema de mensajes:** PlanCode se puede considerar un sistema de mensajes porque permite el intercambio de mensajes de diferente naturaleza entre los usuarios, como es el intercambio de mensajes de texto a través del chat, imágenes cuando se carga un mapa y el intercambio de trazos libres.

3.3. Modelo 3C

Si atendemos al objetivo principal de PlanCode (el diseño de rutas sobre un mapa en situaciones de emergencias), dicha herramienta se puede clasificar como una herramienta de colaboración, pues, como ya se ha comentado, los trazos pueden ser dibujados por cualquier usuario, apoyando al resto de trazos que ya estén dibujados. Del mismo modo, cualquier usuario podría modificar la imagen

del mapa que se está mostrando si lo cree conveniente.

Pero, como PlanCode cuenta con un chat, se puede considerar también una herramienta de comunicación y coordinación.

4. Descripción de los requisitos funcionales

En las siguientes subsecciones, se comentarán sólo aquellos requisitos funcionales que aparecen en [6] y que son cubiertos por PlanCode.

4.1. Participación

- **Métodos para entrar a una sesión:** es un método *predefinido*, pues los usuarios, para poder utilizar esta herramienta deben loguearse.
- **Unión a una sesión:** se permite *entradas posteriores*, pues los usuarios pueden loguearse en cualquier momento.
- **Salidas de una sesión:** en este caso, se *permite salir antes*, pues cualquier usuario puede abandonar la herramienta en cualquier momento.
- **Tamaño del grupo:** PlanCode está diseñado para ser utilizado por *grupos pequeños* (3-5 personas) dentro de una misma sesión.
- **Actividades de trabajo:** las actividades se realizan de manera *conjunta*, pues todos los usuarios interaccionan al mismo tiempo (es decir, de manera síncrona).
- **Formas de comunicación:** PlanCode permite el intercambio de mensajes *textuales* y de *imágenes*.
- **Duración de los elementos:** la duración de los elementos es *no persistente*, ya que no se almacena información producida o modificada en una sesión.
- **Representación de los datos:** los datos que se intercambian tienen una representación *idéntica* en todos los usuarios. Es decir, todos los usuarios reciben los mensajes del chat de la misma forma y se muestran los mismos mensajes a todos, al igual que en la imagen del mapa, la lista de usuarios conectados y los trazos dibujados.

4.2. Interacción

- **Momento de la interacción:** *síncrono*, como ya se comentó en la sección 3
- **Localización:** *distribuida*, pues cada usuario puede interaccionar desde cualquier lugar y momento.
- **Estructura social de la sesión:** la estructura social es *democrática*, ya que un usuario no tiene un rol superior a otro dentro del grupo creado en PlanCode.
- **Convocatoria de los asistentes:** en este caso, la convocatoria podría considerarse como *planeada*, pues la herramienta se va a utilizar cuando ocurra una situación de emergencia.
- **Identificación de los participantes:** los usuarios están *identificados*.
- **Forma de llevar a cabo las actividades:** *simultánea*, pues no hay control de turnos para realizar ninguna acción.

4.3. Coordinación

Como se ha dicho en el último apartado del punto anterior, no existe control de turnos ni aspectos explícitos de coordinación en PlanCode, ya que es una herramienta cuyo principal objetivo es la colaboración.

4.4. Distribución y soporte

- **Almacenamiento de los datos:** el almacenamiento de los datos es *distribuido*, pues no existe un servidor centralizado que almacene y maneje los datos.
- **Arquitectura de implementación:** en lo que se refiere al procesamiento de la información, es una arquitectura *híbrida*, ya que cada cliente procesa y visualiza la información recibida del resto de clientes, pero cuando un cliente inicia sesión, es el "servidor" (el primer cliente que inició sesión) el que le asigna cierta información para su sesión (como el color con el que va a dibujar o la imagen de su avatar).

4.5. Notificación de eventos

- **Conocimiento de uso compartido:** el conocimiento de uso compartido es *consciente*, ya que la herramienta permite una visualización de los usuarios que están conectados, qué usuario envía cada mensaje al chat, etc.
- **Representación gráfica de los participantes:** se utiliza una *lista de usuarios* con avatares según su rol (policía, bombero, etc.) para mostrar los usuarios conectados. Además, cada usuario tendrá asociado un color y los trazos que realice serán de ese color, facilitando la identificación de los trazos que cada usuario dibuja en el área de trabajo compartida.

4.6. Visualización

- **Interfaz de grupo:** la interfaz gráfica de usuario de la herramienta PlanCoDE es de tipo WYSIWIS relajado. La razón por la que se ha elegido la variante *relajada* es para evitar los problemas asociados con las interfaces WYSIWIS estrictas, tales como las guerras de ventanas y la sobrecarga de información. A continuación se profundizará sobre algunas de las características que se han relajado.
 - *Separación del espacio de trabajo:* se ha relajado la barra de menús y las ventanas de uso privado, tales como las ventanas de configuración o carga de planos/imágenes, así como la caja de herramientas. En definitiva, las partes sobre las que se aplica el WYSIWIS es sobre el chat, el log de las acciones, el área de dibujo y la lista de usuarios activos en la sesión. De este modo, los usuarios pueden colaborar para realizar el diseño de las estrategias que estimen oportunas sin perder la intimidad y la privacidad de uso de la aplicación.
 - *Visualización de cursores:* cada usuario tiene su propio cursor privado y ningún otro usuario puede observarlo ni manipularlo. Esto es debido a que PlanCoDE será portada, en un futuro, a dispositivos móviles. Este tipo de pantallas son muy reducidas y no es recomendable permitir que los cursores de los distintos usuarios las invadan, causando confusión.

- *Gestión de la distribución en la pantalla*: cada usuario podrá elegir dónde situar las ventanas *no compartidas* de la aplicación de forma privada.
- *Gestión de la información visualizada*: la aplicación dispone de una ventana de log donde se irán registrando todas las acciones que realizan cada uno de los distintos usuarios en el área de dibujo. De este modo se representa de forma textual la información que se va generando de forma gráfica.
- *Acoplamiento de interfaces de usuario*: el nivel de acoplamiento (ver Figura 4.1) entre las interfaces de usuario de los distintos clientes se clasifica como *medio*. Esto es debido a que hay partes de la interfaz gráfica que son privadas cada usuario, pero hay otras partes como el chat, la lista de usuarios activos y el área de dibujo que son compartidos.

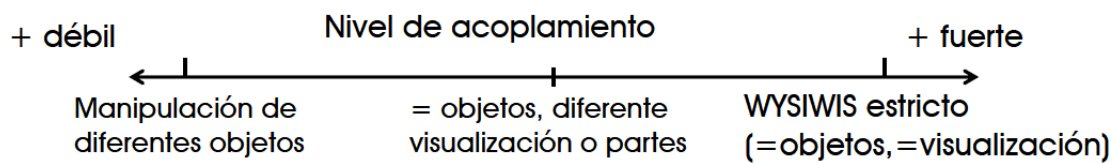


Figura 4.1: Niveles de Acoplamiento de Interfaces de Usuario

- **Transmisión de los datos para mantener las vistas sincronizadas:** existe una transmisión de los datos para que la aplicación correspondiente a cada usuario esté sincronizada.
- **Actualización de la información:** ésta se produce para actualizar el avatar que representa a cada usuario, para actualizar el chat con nuevos mensajes y para actualizar el log de acciones y el área de dibujo cuando un usuario inserta o elimina o un trazo o cuando se carga una imagen de un mapa.

5. Decisiones de diseño

En las siguientes subsecciones se van a detallar las decisiones de diseño que se han tenido en cuenta a la hora de desarrollar e implementar este sistema.

5.1. Lenguaje de programación

Para implementar el sistema, se ha decidido utilizar el lenguaje de programación **Java** ([3]). Además, para realizar la comunicación entre los diferentes clientes de la aplicación, se ha utilizado el *toolkit* de comunicaciones y de intercambio de datos **JSDT** (*Java Shared Data Toolkit* [5]).

Por otra parte, para la realización de la interfaz gráfica, se ha utilizado el paquete **Swing** ([7]) que proporciona Java, junto con el plug-in **Jigloo** ([4]) para el IDE **Eclipse** ([2]).

5.2. Diseño multicapa

El sistema *groupware* ha sido desarrollado siguiendo un diseño multicapa, para separar aspectos de la lógica de negocio de aspectos de presentación y de comunicaciones. De este modo, el sistema es extensible y reutilizable, pudiéndose portar en un futuro a una aplicación móvil, ya que sólo habría que modificar la capa de presentación y la de comunicaciones, de acuerdo al API de comunicaciones que se desee utilizar en este tipo de dispositivos.

Así, la aplicación cuenta con las capas de dominio, presentación y comunicaciones. A continuación se enumera el propósito de cada uno de esas capas (o paquetes).

- **Dominio:** en esta capa se incluyen todas las clases necesarias para modelar el conocimiento y para controlar la lógica del sistema PlanCode. Así, en el paquete *dominio.conocimiento* se incluyen clases que modelan información necesaria en el sistema, tales como la información de un trazo y la información acerca de un usuario de la aplicación (su nombre de usuario, su rol, etc.). En el paquete *dominio.control* se incluyen clases de control de la propia aplicación, como la clase que controla los colores asignados a los usuarios o el controlador de la aplicación, que inicializa la sesión, los canales, los consumidores, etc. El contenido de esta capa se muestra en la Figura 5.1 y se puede observar con más detalle en el diagrama de clases de Visual Paradigm ([8]) llamado *dominio*.
- **Comunicaciones:** en esta capa se incluyen las clases necesarias para poder comunicar y enviar datos entre los diferentes usuarios de PlanCode utilizando el *toolkit* JSDT. Así, en el paquete *comunicaciones* se incluyen las clases que modelan los consumidores de cada uno de los canales con los que cuenta la aplicación (ver apartado 5.3.2), además de los datos de una sesión (IP y puerto de comunicación). En el paquete *comunicaciones.EventosCanales* se encuentran todas las clases que sirven para modelar los eventos que se producen cuando se envía información a través de un canal de JSDT. Dichos eventos recogen la información enviada (mensaje de chat, trazo dibujado, etc.). Además, cada evento tiene su propio manejador (*listener*) para que el consumidor del canal donde se produce dicho evento pueda atenderlo y procesarlo de manera correcta. El contenido de esta capa se muestra en la Figura 5.2 y se puede observar con más detalle en el diagrama de clases de Visual Paradigm llamado *comunicaciones*.

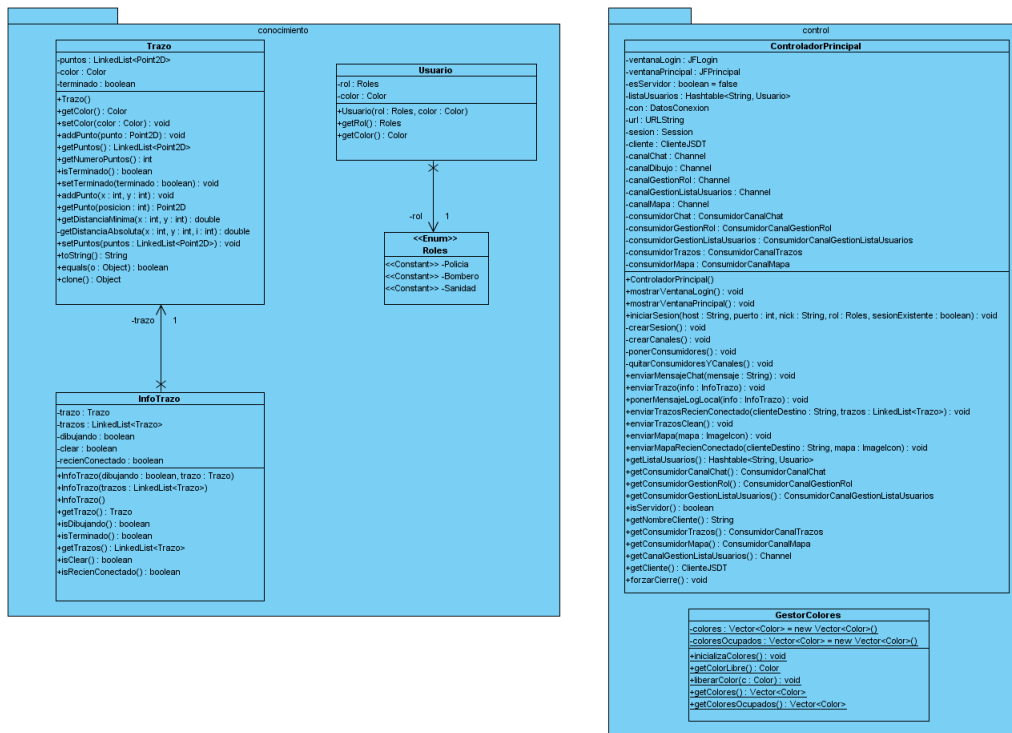


Figura 5.1: Diagrama de clases de la capa de dominio

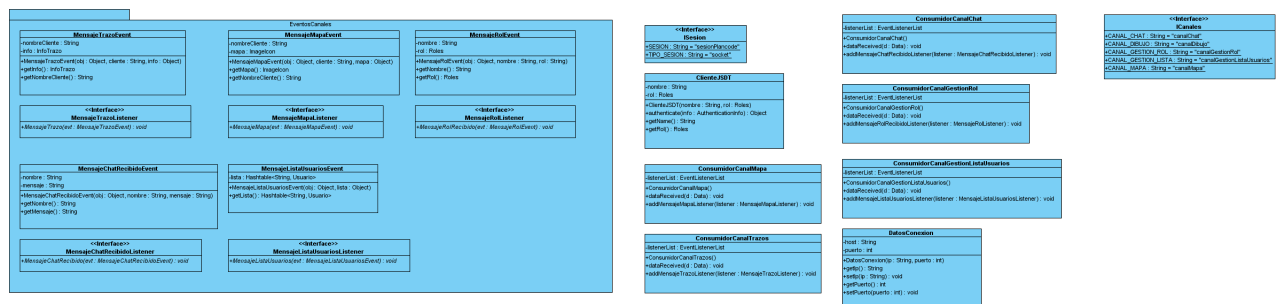


Figura 5.2: Diagrama de clases de la capa de comunicaciones

- **Presentación:** todas las clases relacionadas con la interfaz gráfica de usuario se encuentran en este paquete. De este modo, se incluye en esta capa las clases necesarias para mostrar la ventana de login de la aplicación, la ventana principal y el componente *canvas* que permite dibujar trazos. Además, en el paquete *presentación.auxiliares* se incluyen clases necesarias para la visualización de información en la interfaz gráfica, tales como diálogos de error, interfaces para capturar los eventos de ratón al dibujar, clases para cargar una imagen en un panel, etc. El contenido de esta capa se muestra en la Figura 5.3 y se puede observar con más detalle en el diagrama de clases de Visual Paradigm llamado *presentacion*.

En la Figura 5.4 se muestra el diseño de clases del sistema PlanCode completo, donde se muestra todas las clases de las capas y las relaciones entre ellas. Debido a su extensión, se recomienda consultar el diagrama de clases de Visual Paradigm llamado *DiagramaPlanCode*.



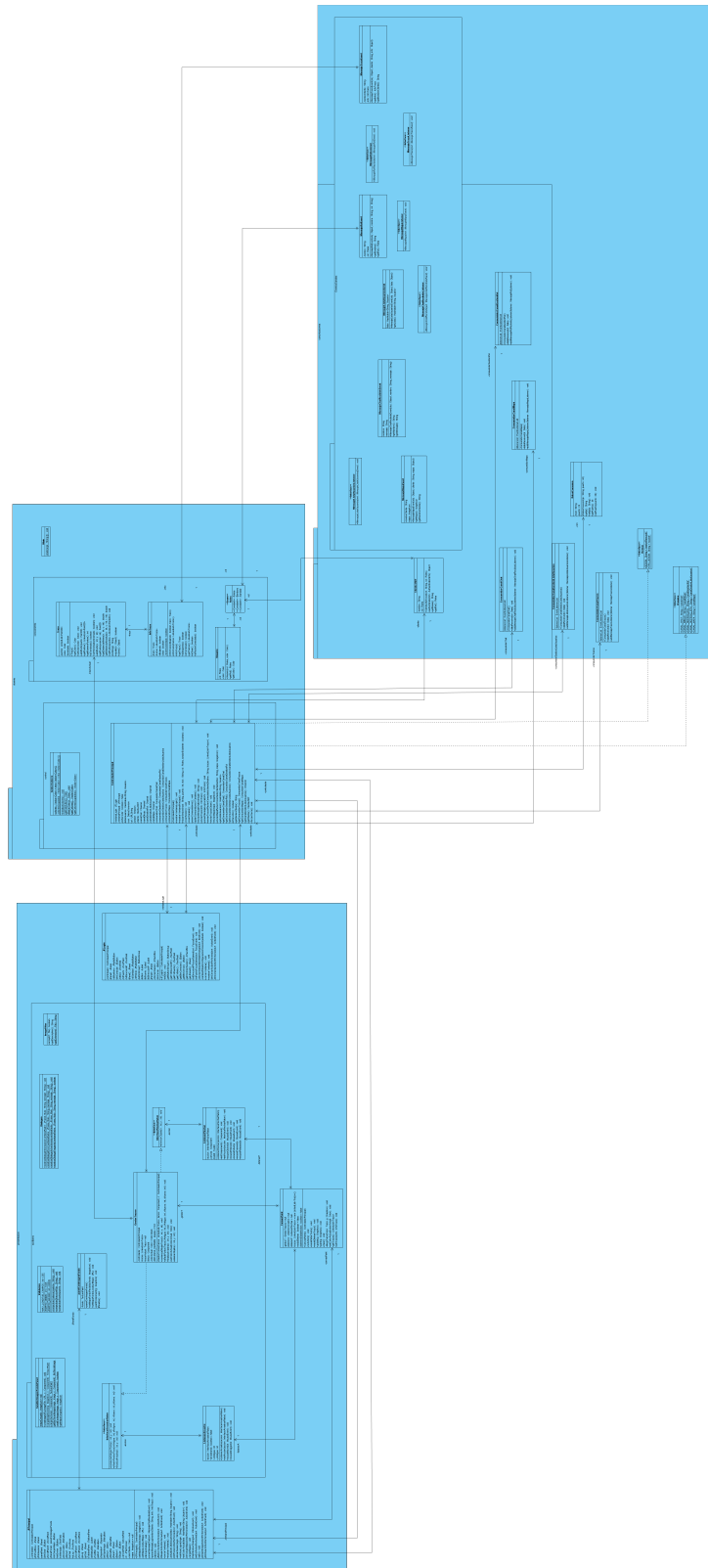


Figura 5.4: Diagrama de clases del sistema PlanCode

5.3. Desarrollo del sistema

5.3.1. *Arquitectura de PlanCode*

PlanCode, como se mencionó en la sección 2, sigue una arquitectura cliente-cliente, donde el primer cliente que va a utilizar el sistema se convierte en servidor, asegurando la disponibilidad de la sesión para que otros clientes puedan conectarse.

De este modo, este cliente especial tiene la responsabilidad de crear la sesión y de controlar ciertos aspectos cuando otros clientes se conectan a su sesión (es decir, a su dirección IP). Dichos aspectos son:

- Inicializar la lista de colores disponibles.
- Recibir el rol cuando otro cliente se conecta, asignándole un color y estableciendo el avatar correspondiente.
- Añadir a la lista de usuarios conectados el cliente que se conecta a la aplicación, con su avatar y su color, enviando esa información a los clientes ya conectados para que refresquen su lista de usuarios.
- Enviar al cliente que acaba de iniciar sesión el mapa que hubiese ya cargado y los trazos que hubiese ya dibujados, para sincronizar la vista de ese cliente con el resto.

5.3.2. *Canales implementados*

Como ya se comentó anteriormente, se ha utilizado el *toolkit* JSMT para la comunicación y envío de información entre clientes. Los canales que se han utilizado para el paso de información han sido:

- **canalChat**: utilizado para enviar a los clientes los mensajes que se escriben en el chat.
- **canalDibujo**: utilizado para enviar a los clientes los trazos que se dibujan o se borran del área de trabajo compartida.
- **canalMapa**: utilizado para enviar a los clientes la imagen que se ha cargado.
- **canalGestiónRol**: utilizado para enviar al cliente que actúa como "servidor" el rol del cliente que acaba de iniciar sesión, para que el servidor pueda, en función de ese rol, cargar la imagen del avatar correspondiente y asignarle un color.
- **canalGestiónListaUsuarios**: utilizado para enviar a los clientes la lista de usuarios conectados actualizada tras el inicio de sesión de un nuevo cliente. La lista de usuarios contiene el avatar del cliente (según su rol), el nombre de usuario con el que inicia sesión y su color asignado.

5.3.3. *Prototipo de la interfaz gráfica de usuario*

La interfaz gráfica de PlanCode ha sido desarrollada siguiendo el prototipo mostrado en la Figura 5.5.

Los elementos que componen dicha interfaz son:

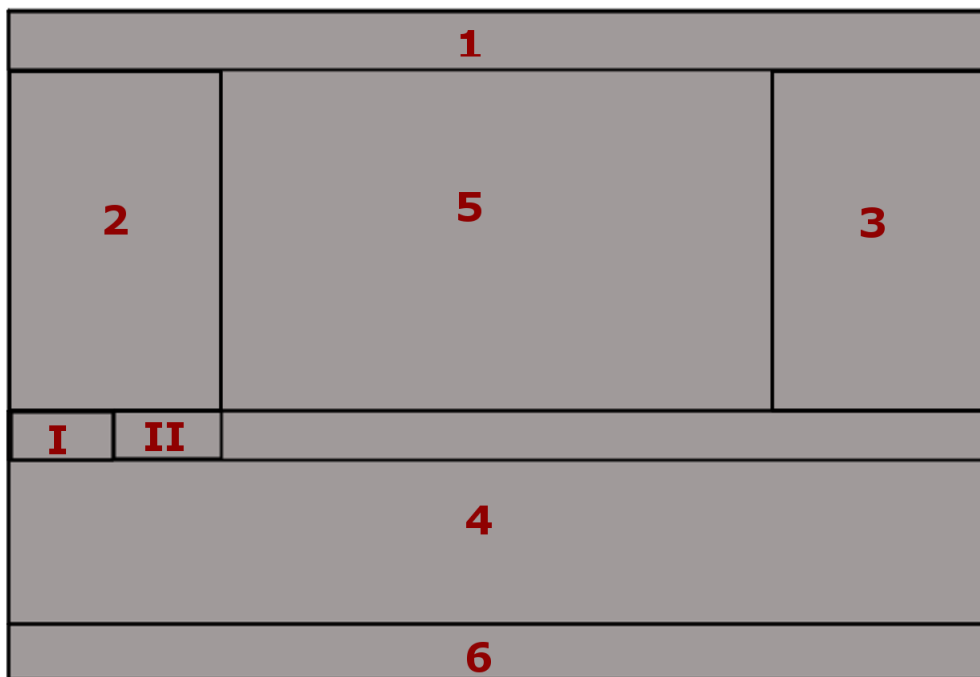


Figura 5.5: Prototipo de la interfaz gráfica de PlanCoDE

1. **Barra de herramientas:** en este elemento se mostrarán los diferentes menús de la aplicación.
2. **Caja de herramientas:** aquí se mostrarán las herramientas necesarias para poder dibujar y eliminar trazos sobre las imágenes de los mapas.
3. **Lista de usuarios:** en este elemento se muestran los usuarios que están utilizando la aplicación.
4. **Terminal:** en la primera pestaña de este elemento se muestran los mensajes que se envían al chat, y en la segunda, se muestra todo el *log* de acciones que realizan los usuarios.
5. **Área de trabajo:** aquí se muestra la imagen del mapa seleccionado y se establece el *canvas* para poder dibujar y eliminar trazos.
6. **Barra de estado:** este elemento indica el usuario que ha iniciado sesión y con que rol.

Como se puede apreciar en la Figura 5.5, la interfaz de usuario se ha diseñado siguiendo los principios de WYSIWIS relajado que se comentaron en la sección 4.

5.3.4. Gestión de trazos

La principal funcionalidad de PlanCode es el diseño de rutas sobre mapas para situaciones de emergencia. Es por tanto que la gestión de los trazos que los usuarios pueden dibujar sobre los mapas es la parte más importante en el diseño de este sistema *groupware*.

Para poder dibujar trazos sobre una imagen (en este caso, una imagen de un mapa), no se puede utilizar el componente *Canvas* que proporciona Java, ya que dicho componente no es transparente.

Por ello, se decidió implementar un *canvas* personalizado a partir de un *JPanel*. De este modo, la imagen del mapa seleccionado se carga sobre un *JPanel* al que se le añade dicha imagen como fondo, y sobre este panel se coloca el *canvas*.

En lo que concierne a la visualización de los trazos, éstos se visualizan en tiempo real según el usuario los va dibujando. Así, cada vez que el usuario arrastra el ratón para dibujar un trazo, se toman las coordenadas por la que se está moviendo el ratón y se van añadiendo esos puntos al trazo y visualizándolos en el área de trabajo. Además, toda esa información del trazo se envía al resto de clientes, que también visualizan los trazos que otro cliente está realizando en tiempo real. De este modo, un trazo se representa como una lista enlazada de puntos 2D.

De forma análoga, cuando un usuario desea borrar un trazo, se captura la posición donde ha pinchado y se borra el trazo de esa posición (o de una muy próxima) y se refresca el área de trabajo compartida, borrando también ese trazo en el resto de clientes.

Para poder eliminar un trazo, ha sido necesario implementar un pequeño algoritmo de distancia para comprobar la distancia que hay entre la posición donde el usuario pincha con el ratón y el trazo más cercano. Esto es así porque no se tiene la suficiente precisión para poder hacer clic con el ratón justo encima de un punto del trazo que se desea eliminar, por lo que hay que obtener con este algoritmo de distancias el trazo más cercano al punto donde el usuario ha pinchado.

Para terminar, cabe destacar que los eventos del ratón que se utilizan para dibujar o eliminar trazos están encapsulados dentro de manejadores y que las acciones que se pueden hacer en el área de trabajo (dibujar y eliminar trazos) se definen como interfaces (ver subsección 5.2). De este modo, se podría extender el sistema con nuevas funcionalidades en el área de trabajo sólo modificando los manejadores y extendiendo las interfaces de las acciones permitidas.

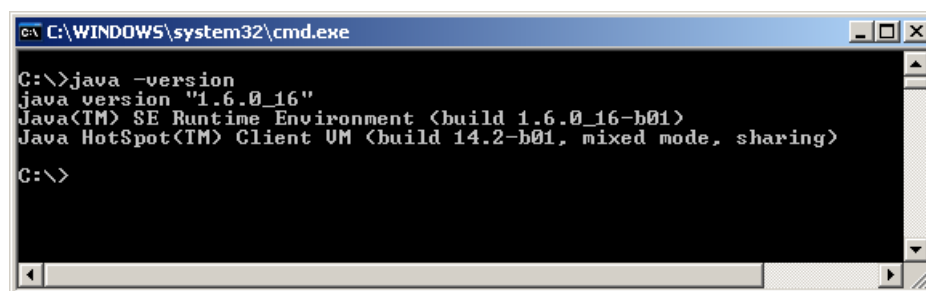
6. Manual de usuario

6.1. Requisitos del sistema

El sistema PlanCode se ha desarrollado bajo el lenguaje de programación Java para hacer posible su portabilidad entre distintos sistemas, siempre y cuando éstos tengan instalada una máquina virtual de Java. De este modo, las aplicaciones son compatibles tanto en entornos basados en Windows 98/NT/XP/Vista/Seven como en entornos GNU/Linux.

6.1.1. Entorno de Java

Para comprobar si ya dispone de una instalación del entorno de ejecución de Java en el equipo y que está configurado correctamente, abra una ventana de comandos (*Inicio -> Ejecutar -> cmd*) y escriba "java -version". Debe obtener un resultado similar al de la Figura 6.1.



```
C:\WINDOWS\system32\cmd.exe

C:\>java -version
java version "1.6.0_16"
Java(TM) SE Runtime Environment (build 1.6.0_16-b01)
Java HotSpot(TM) Client VM (build 14.2-b01, mixed mode, sharing)

C:\>
```

Figura 6.1: Cómo averiguar la versión de Java

Si aparece un mensaje de error, pruebe lo siguiente:

- Compruebe si tiene instalado Java. Para ello compruebe que exista el directorio Java en *C:\Archivos de programa* (típicamente). Si lo está, configúrelo como se indica más abajo.
- Si no existe, vaya a la siguiente dirección de Internet y descargue la última versión del Java SE Development Kit (JDK) [1]

Una vez descargado, ejecute el archivo, elija el directorio de instalación y siga los pasos para completar la instalación. Tras instalarlo, debe configurarlo. Para ello vaya a *Inicio* y haga clic derecho en *Mi PC*. En las opciones, elija la pestaña de *Opciones Avanzadas* y haga clic en *Variables de entorno*. En la nueva ventana que aparece, busque la variable *Path* y añada al final *;C:\Archivos de programa\Java\jdk1.x.x_xx\bin*, donde "x" hace referencia a la versión que tiene instalada de Java.

Ese es el directorio de instalación por defecto de Java. Si en la instalación eligió otro diferente, debe introducir el que eligió al instalarlo.

Tras esto, pruebe de nuevo a ejecutar "java -versión" en una ventana de comandos.

6.2. Guía de uso

Al ejecutar el fichero JAR correspondiente a la aplicación, obtenemos una ventana como la que se muestra en la Figura 6.2. Si la describimos de arriba abajo, se aprecian los siguientes elementos:

- Una caja de texto para introducir el nombre de usuario del cliente.
- Unos botones para elegir el rol del usuario en la aplicación.
- Un botón para elegir si se desea iniciar una nueva sesión o unirse a una sesión ya iniciada. En este último caso, se debe introducir la dirección IP del cliente que inició la sesión por primera vez, además del puerto de escucha.

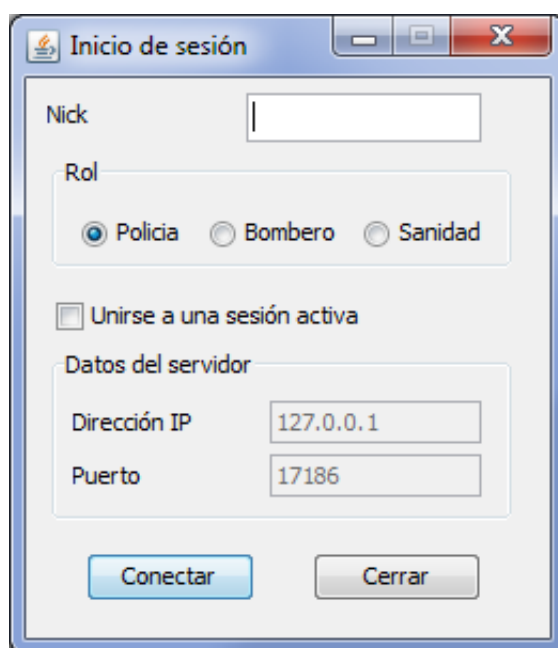


Figura 6.2: Ventana de inicio de sesión de PlanCode

Al iniciar sesión correctamente, se abrirá la ventana principal de PlanCode, mostrada en la Figura 6.3. Como se puede observar, en la lista de usuarios situada en la derecha, aparecen los usuarios conectados actualmente a una sesión. A continuación, se muestran las distintas funcionalidades que ofrece esta herramienta:

- **Chat:** situado en la parte inferior de la ventana, en la primera ventana. Aquí se mostrarán los mensajes que los usuarios envíen, como se muestra en la Figura 6.4.
- **Cargar imagen:** se puede cargar una imagen en el botón situado en la parte superior o en la opción correspondiente del menú "Archivo". Cuando un usuario cargue una imagen, automáticamente se enviará dicha imagen al resto de usuarios conectados y a los futuros clientes que se conecten, para mantener sincronizada el área de trabajo, como se muestra en la Figura 6.5. Además, como se observa en dicha figura, se actualiza el log de acciones en consecuencia.
- **Dibujar trazo:** un cliente puede dibujar trazos libremente sobre la imagen cargada (aunque se puede dibujar también sin tener una imagen cargada) seleccionando el botón con el icono del lápiz, mostrándose dichos trazos en todos los clientes conectados. Además, los futuros clientes que se conecten también recibirán esos trazos para mantener el sincronismo de la vista

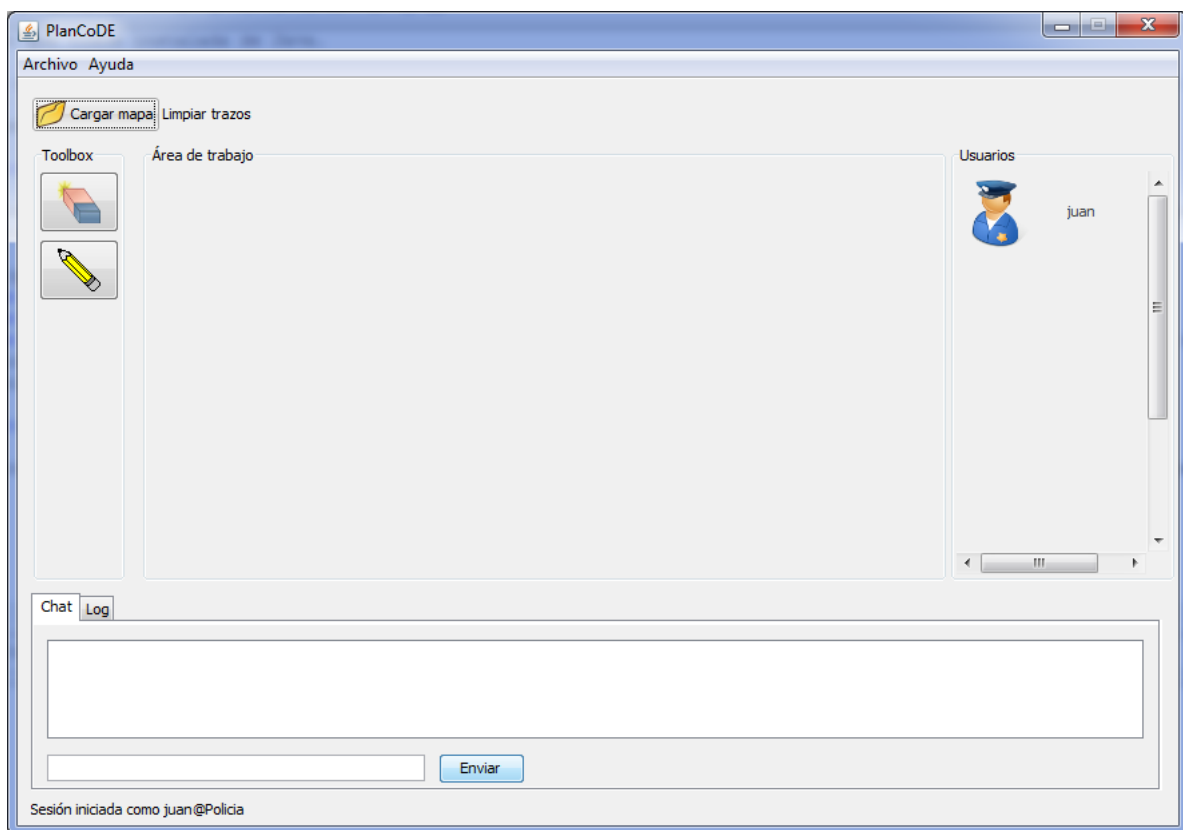


Figura 6.3: Ventana principal de PlanCode

compartida. En la Figura 6.6 se muestra un ejemplo de como cada cliente puede dibujar trazos, con su color correspondiente.

- **Eliminar trazos:** se puede eliminar un trazo, seleccionando el botón con el icono de la goma de borrar y pinchando en el trazo que se desea borrar. El resto de clientes se actualizarán en consecuencia, como muestra la Figura 6.7.
- **Limpiar trazos:** se pueden limpiar todos los trazos dibujados al pulsar sobre el botón correspondiente, o al cargar una imagen.
- **Cerrar sesión:** para salir del sistema, basta con darle a la opción correspondiente del menú "Archivo" o cerrar la ventana. Cuando un usuario se desconecta, se notifica en el log y se refresca la lista de usuarios conectados, como se muestra en la Figura 6.8

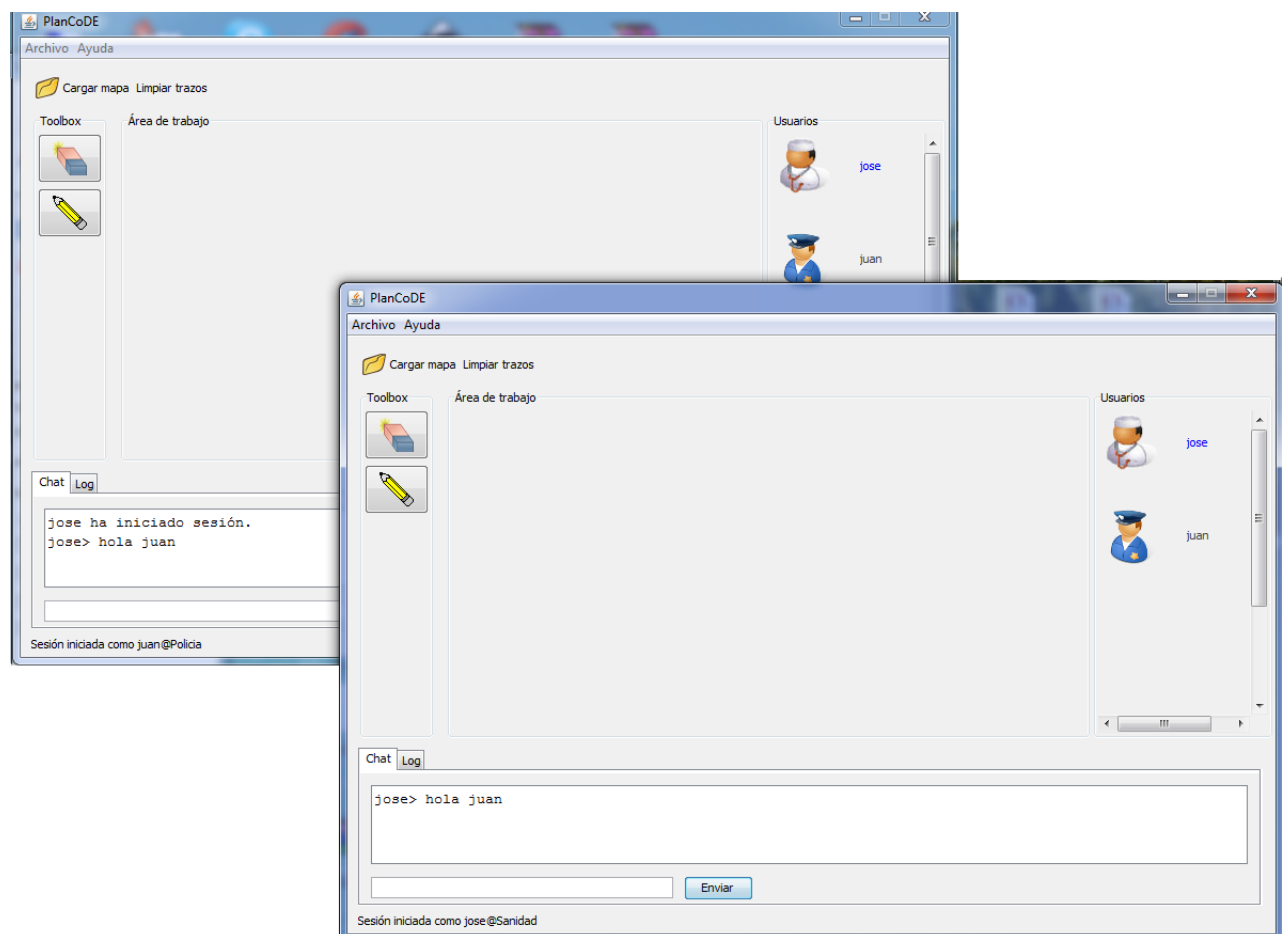


Figura 6.4: Ventana principal de PlanCode al enviar un mensaje al chat

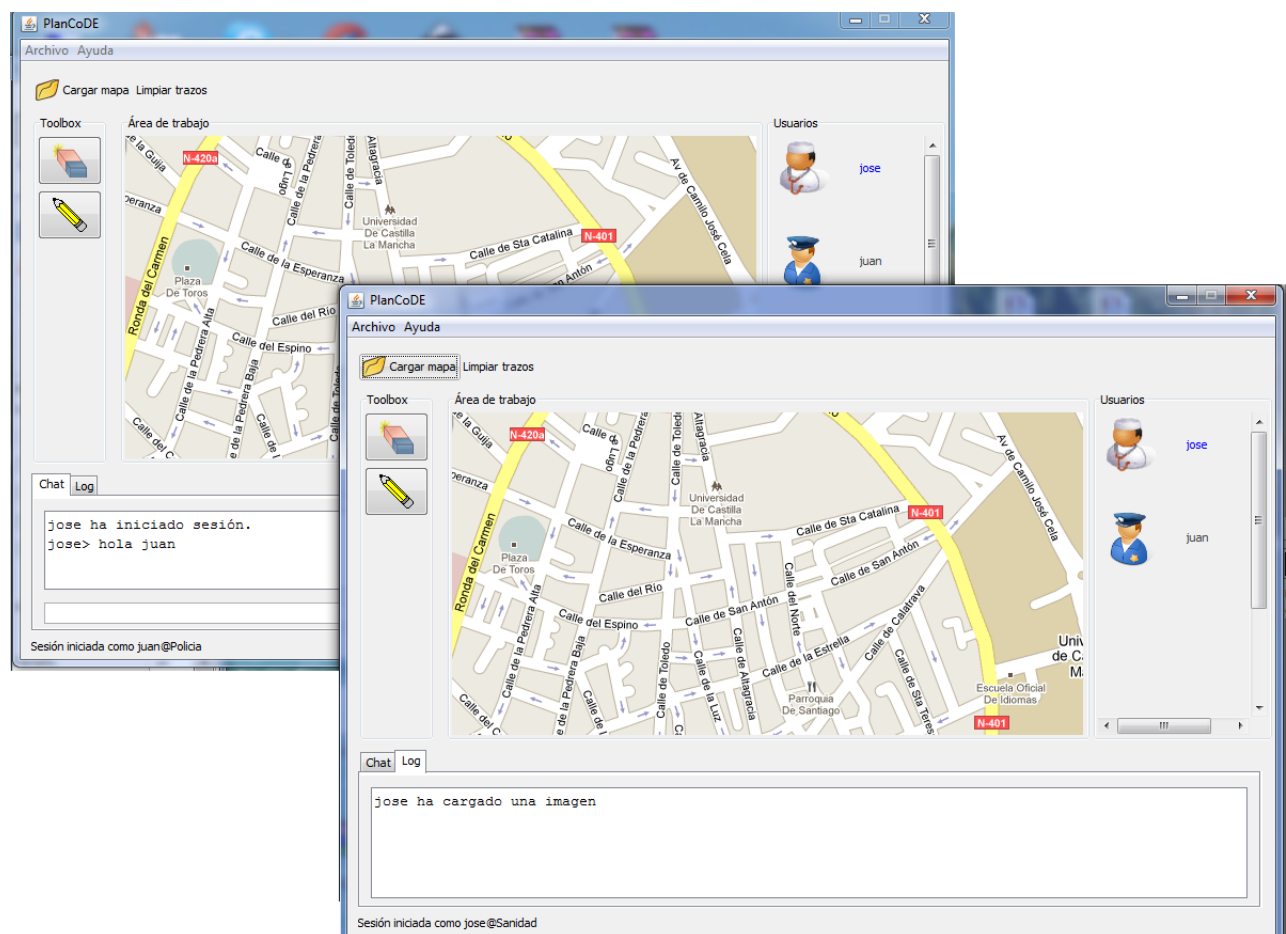


Figura 6.5: Ventana principal de PlanCode al cargar una imagen

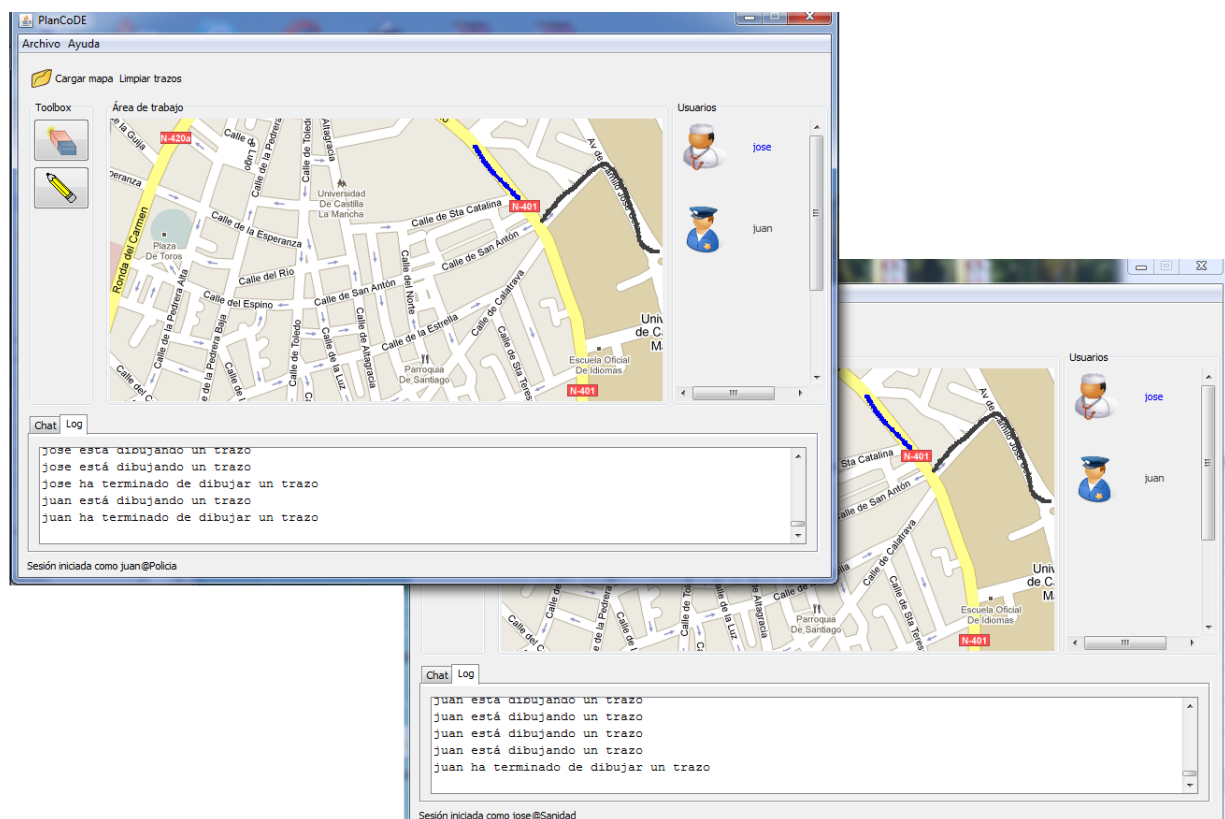


Figura 6.6: Ventana principal de PlanCode al dibujar trazos

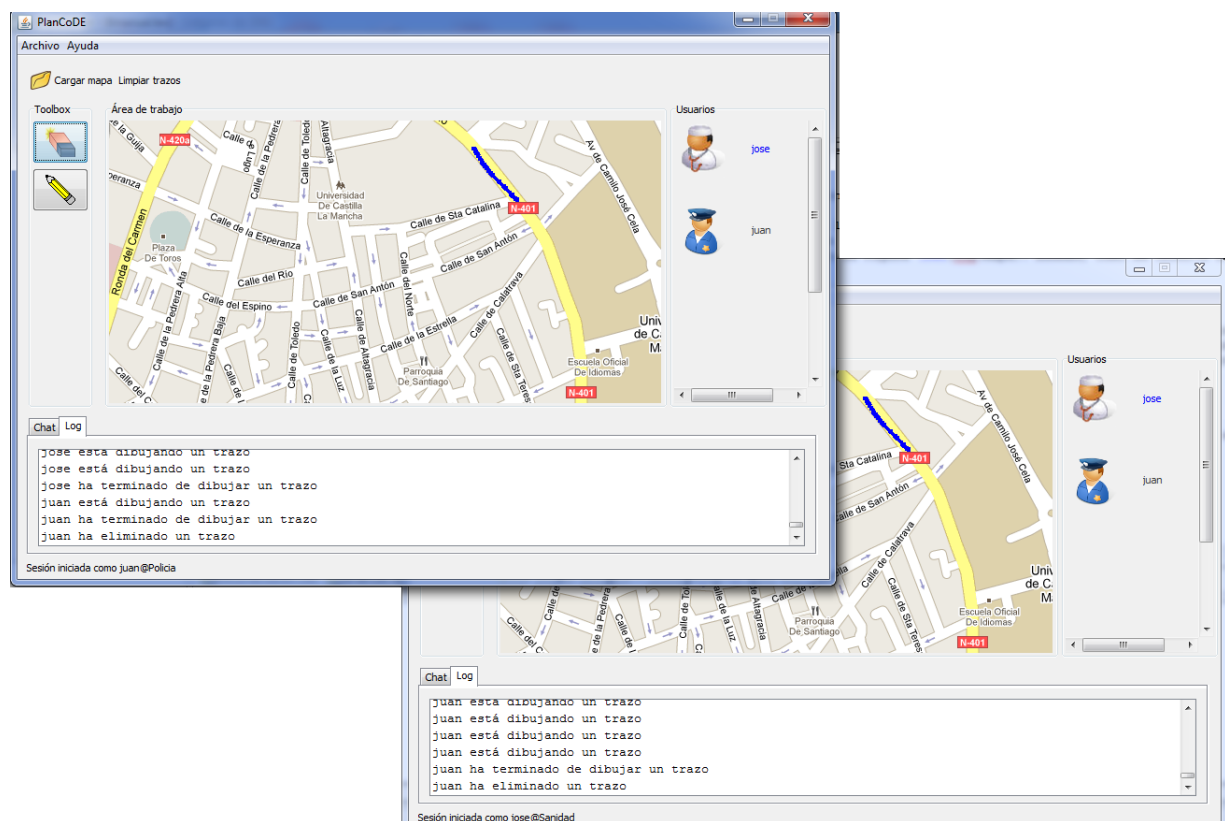


Figura 6.7: Ventana principal de PlanCode al borrar trazos

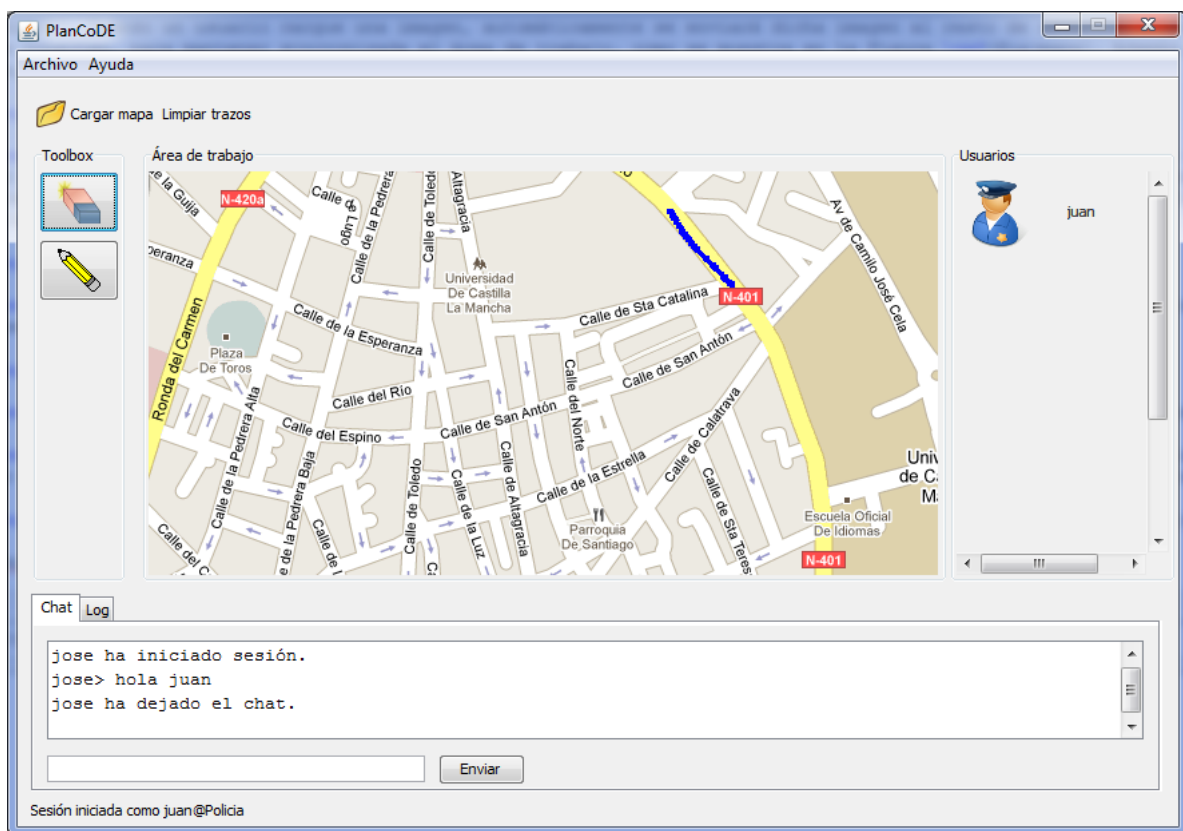


Figura 6.8: Ventana principal de PlanCode cuando se desconecta un usuario

7. Informe de división de trabajo

El sistema ha sido desarrollado en conjunto por los dos integrantes del grupo, por lo que el grado de participación de cada uno es del 50 %. La mayoría de la práctica se ha desarrollado de manera conjunta y colaborativa, sólo diviendo el trabajo para desarrollar por separado secciones de la documentación y algunas funciones del código.

Referencias

- [1] Página Web para descargas de Java. <http://java.sun.com/javase/downloads/index.jsp>.
- [2] Eclipse Home Page. <http://www.eclipse.org/>.
- [3] Java Home Page. <http://java.sun.com/javase/>.
- [4] Plug-in Jigloo. http://cloudgarden1.com/jigloo_444.zip.
- [5] JSDT Home Page. <https://jsdt.dev.java.net/>.
- [6] Ana I. Molina. Sistemas para la Colaboración 09-10, Tema 2: Groupware.
- [7] Java Swing. <http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/package-summary.html>.
- [8] Visual Paradigm Home Page. <http://www.visual-paradigm.com/>.