INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

REDES DE COMPUTADORAS

PROF. AXEL ERNESTO MORENO CERVANTES

GRUPO: 3CM7

PRÁCTICA 3 – CHAT MULTICAST

ALEJANDRO DE JESÚS ZEPEDA FLORES

10 de mayo de 2020

Prácticas: https://drive.google.com/drive/folders/12lUUJImgU72oXJfukVcbrB8OjSsMSkb2?usp=sharing

Práctica 3: https://drive.google.com/open?id=1fq7fYLCbXQpww6fS1ziLMeFKn5D6O\_nl

OBJETIVOS

Implementar un servicio de Chat que permita a los usuarios:

• Enviar y recibir mensajes públicos o privados.

• Enviar y recibir archivos vía multicast.

• Enviar emojis o gifs.

INTRODUCCIÓN

En transmisión de datos, la tecnología multicast representa un servicio de red en el cual un único flujo de datos, proveniente de una fuente determinada, se puede enviar simultáneamente a varios receptores interesados. Es función de la infraestructura de red transportar este flujo de datos, replicándolo cuando sea necesario, a todos los receptores que registren interés en recibir los mismos.

El multicast está orientado a aplicaciones del tipo “uno a muchos” (1 a N) o “muchos a muchos” (N a M). En estos casos, presenta claras ventajas cuando se lo compara con los mecanismos de transmisión unicast y broadcast.

Con multicast, la fuente de la transmisión envía una única copia de los paquetes hacia una dirección de grupo multicast. La infraestructura de red replica estos paquetes de forma inteligente, encaminando los datos de acuerdo con la topología de receptores interesados en esa información. La ventaja principal de la transmisión de datos multicast es una clara reducción en la carga de la red.

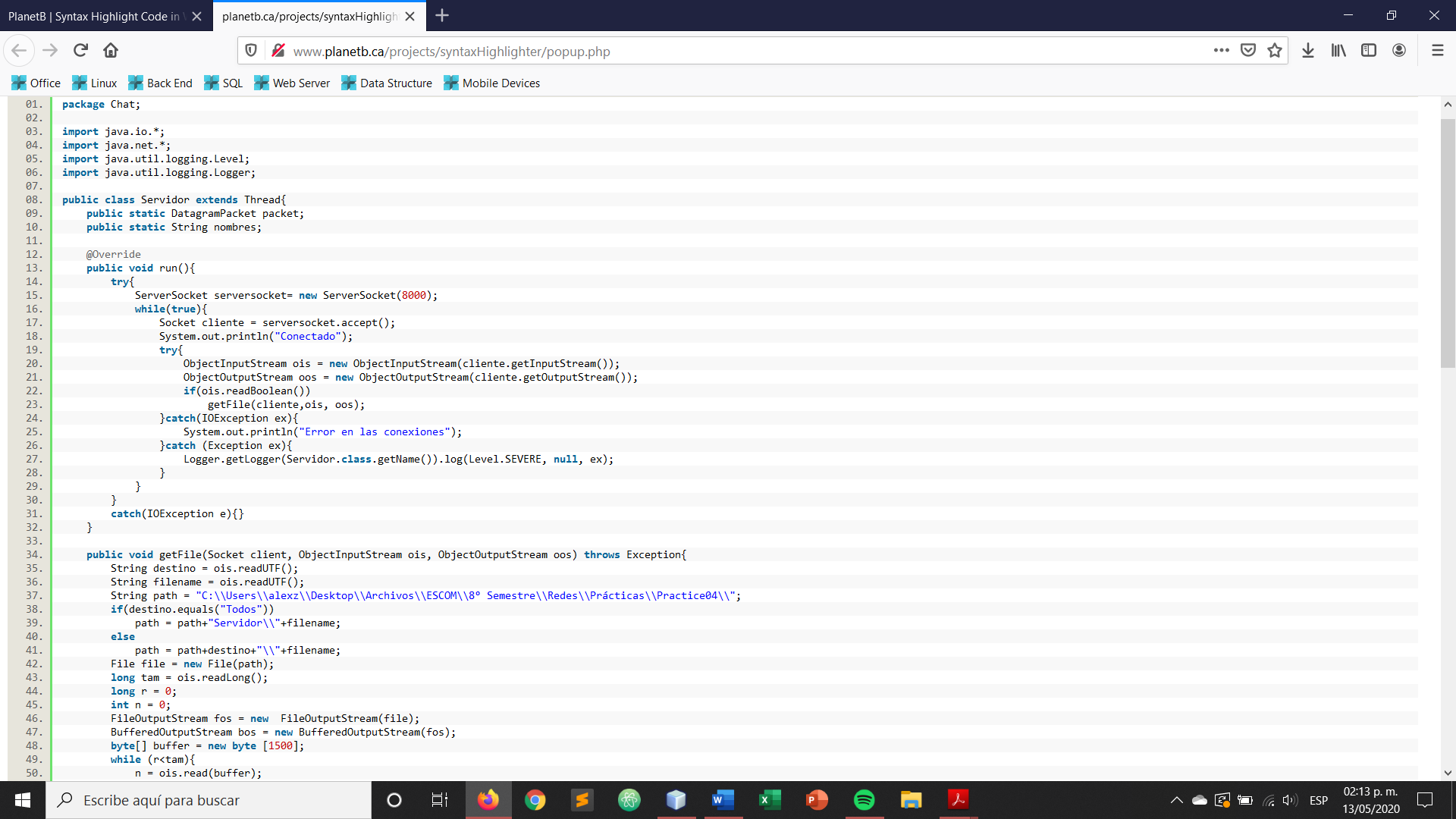
Entre las diversas aplicaciones que se pueden beneficiar con el uso de multicast encontramos: videoconferencia; aprendizaje a distancia; distribución de archivos y software, conciertos en vivo, actualización de bases de datos, juegos distribuidos, streaming, etc.

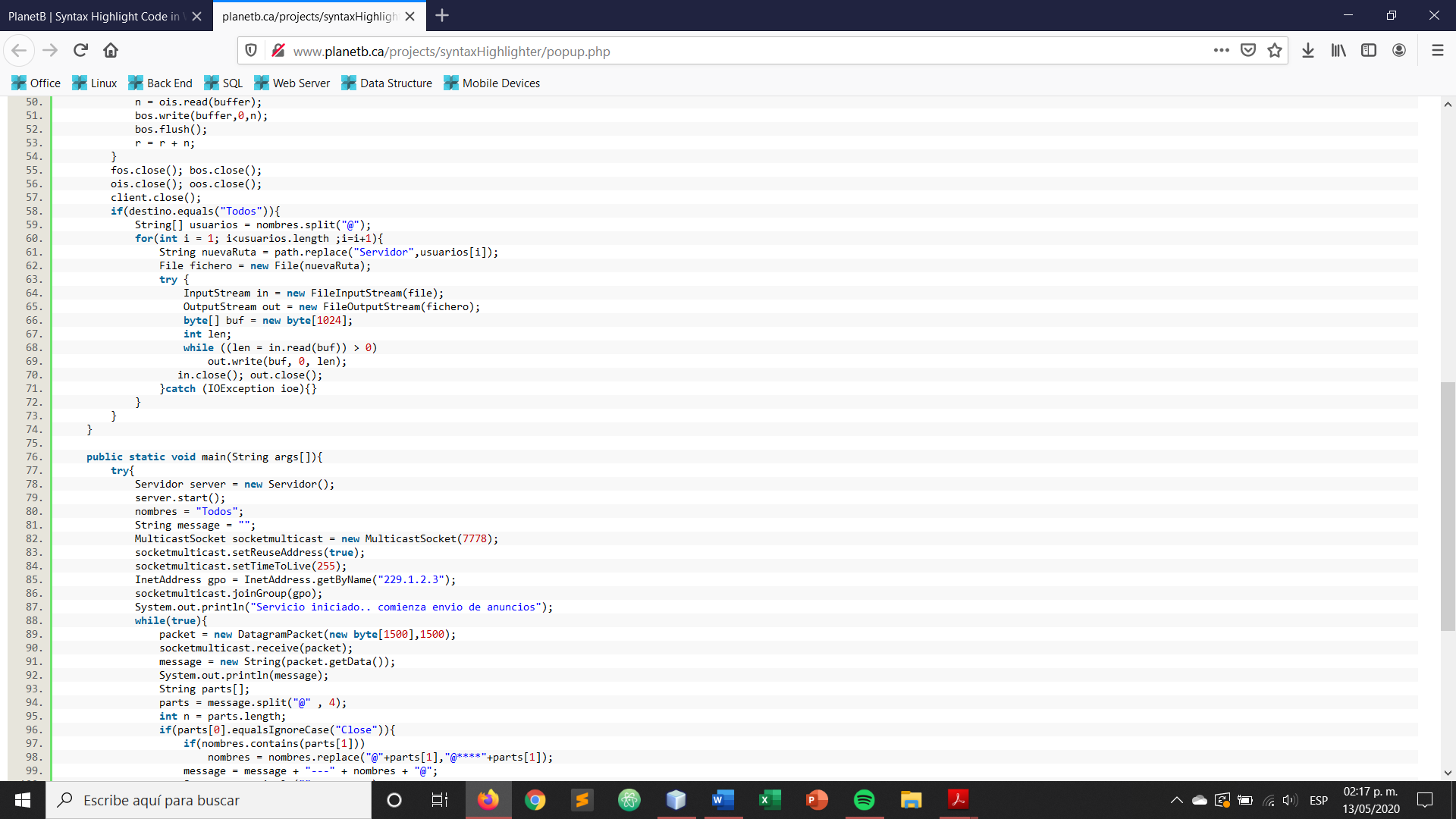
DESARROLLO

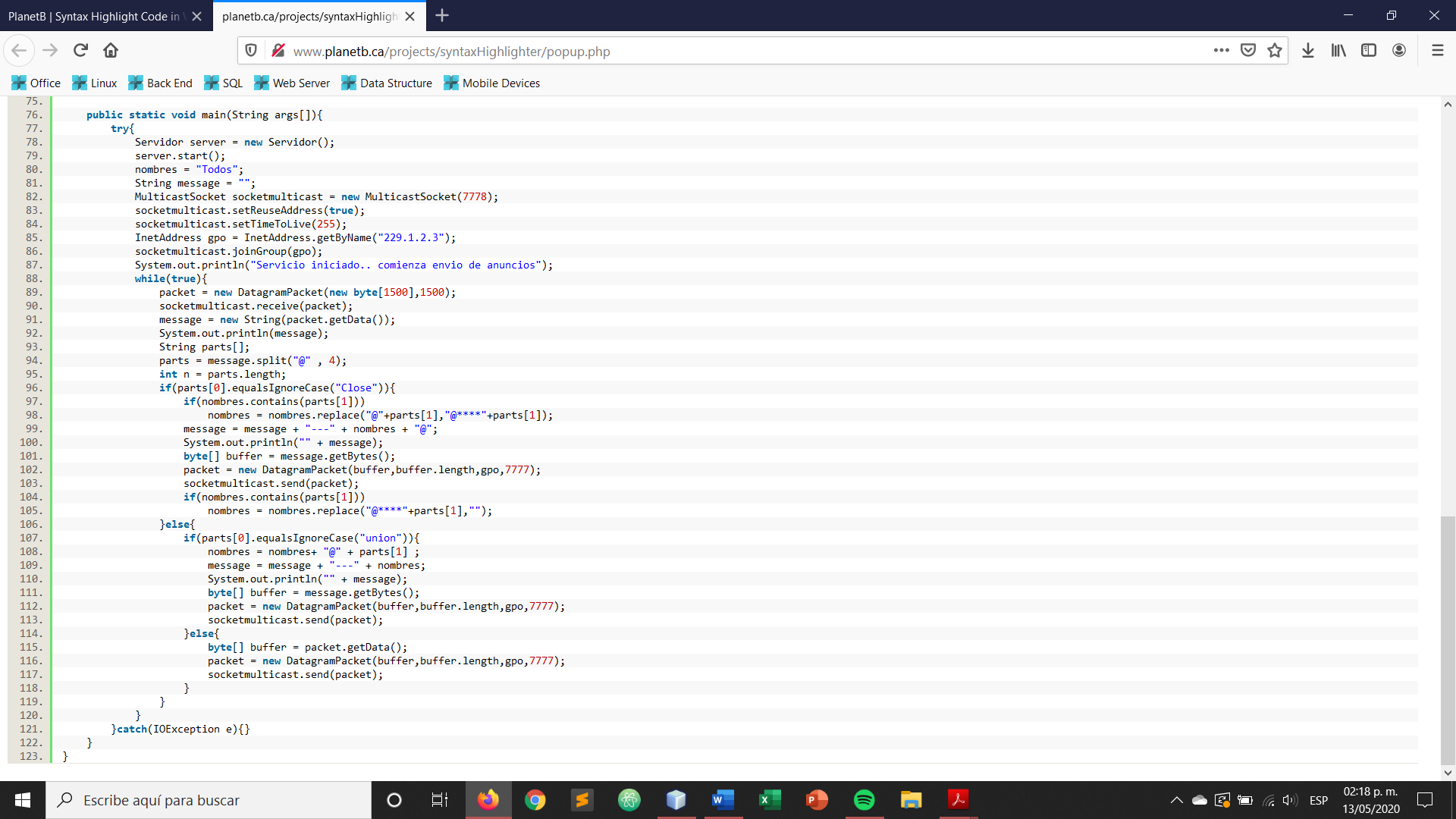
Iniciamos con la programación del Servidor, ya que este es el que estará enviando y recibiendo los datos y los direcciona con los clientes. Realmente las únicas funciones que se pueden observar no realizan un funcionamiento complicado.

* **Run():** como la clase Servidor extiende Thread, es posible implementar este método para poder tener en constante actualización de cambios. Básicamente, en este método se utiliza para poder enviar y recibir los archivos.
* **getFile(Socket client, ObjectInputStream ois, ObjectOutputStream oos):** como este método es llamado en el método run, reutilizamos el socket server para lograr la conexión con el cliente, así como los flujos de datos.
  + En el método **server.accept()**se puede realizar una espera de conexión por parte del cliente mediante el método accept(). Hay que decir, que este método es de bloqueo, el proceso espera a que se realice una conexión por parte del cliente para seguir su ejecución. Una vez que se ha establecido una conexión por el cliente, este método devolverá un objeto tipo Socket, a través del cual se establecerá la comunicación con el cliente.
  + El método **getInputStream()** de la clase Java Socket devuelve una secuencia de entrada para el socket dado. Si cierra el InputStream devuelto, cerrará el socket vinculado.
  + El método **getOutputStream()** de la clase Java Socket devuelve una secuencia de salida para el socket dado. Si cierra el OutputStream devuelto, cerrará el socket vinculado.

A continuación, se muestra el código implementado en el Servidor.





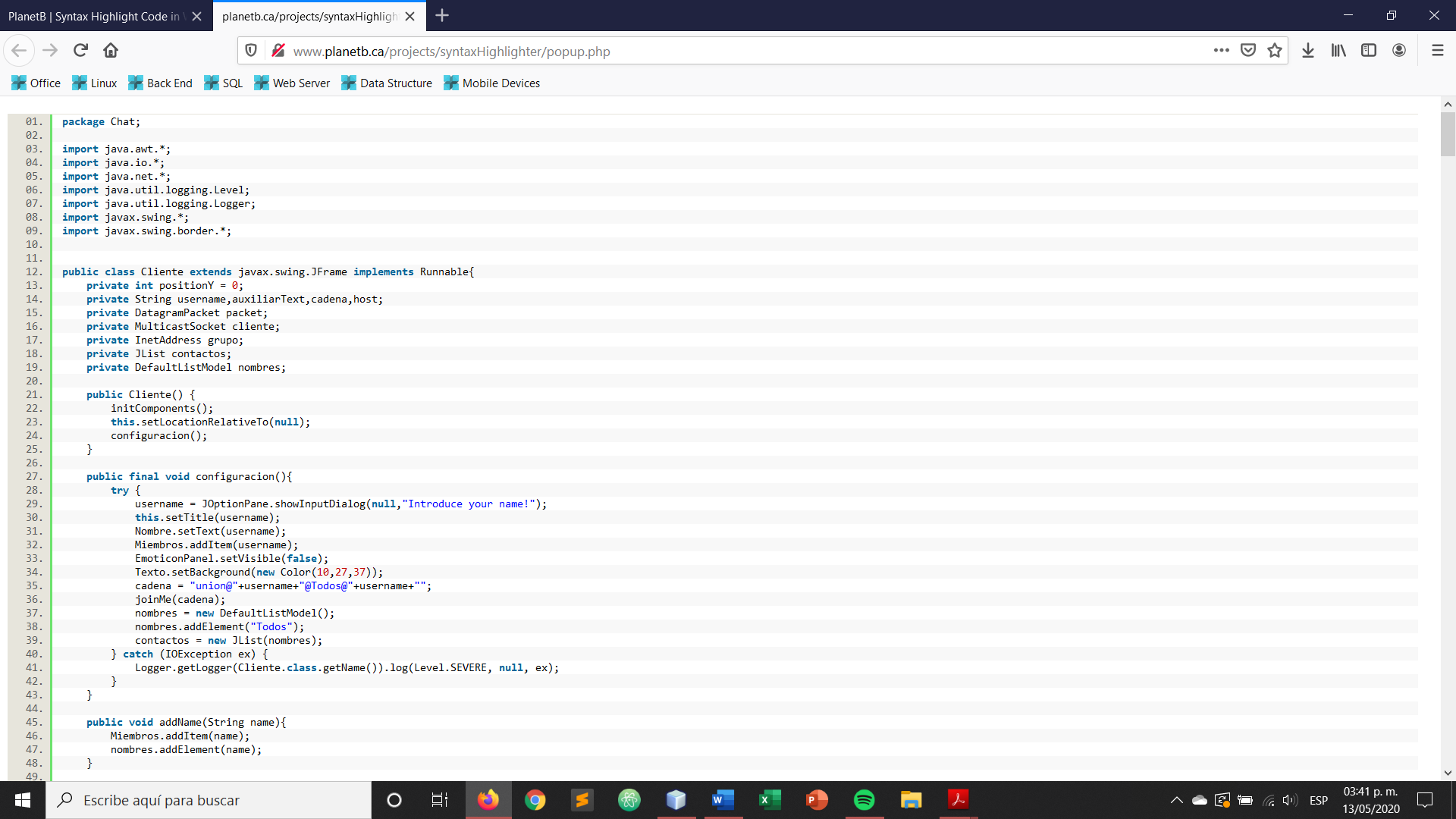


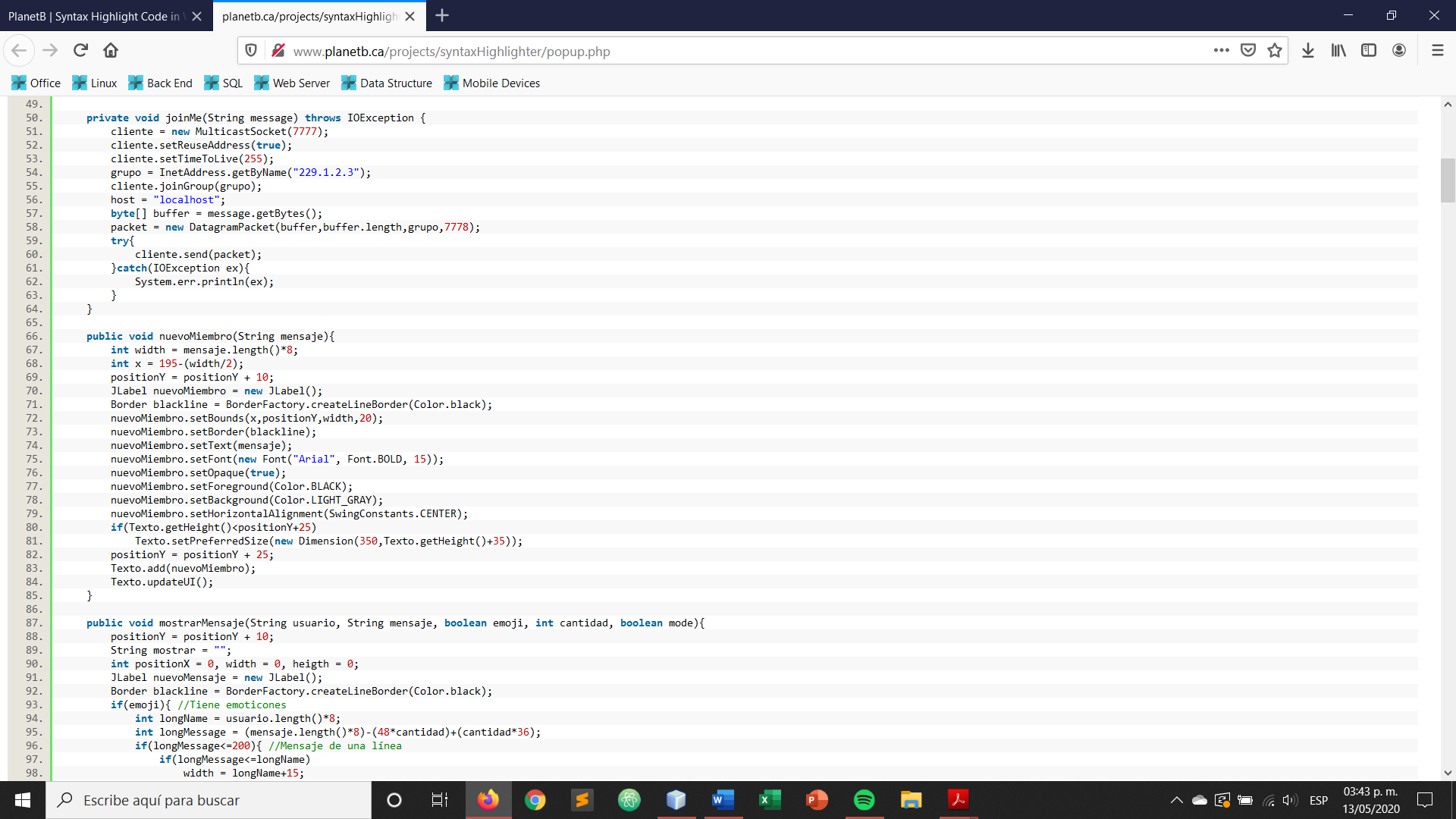
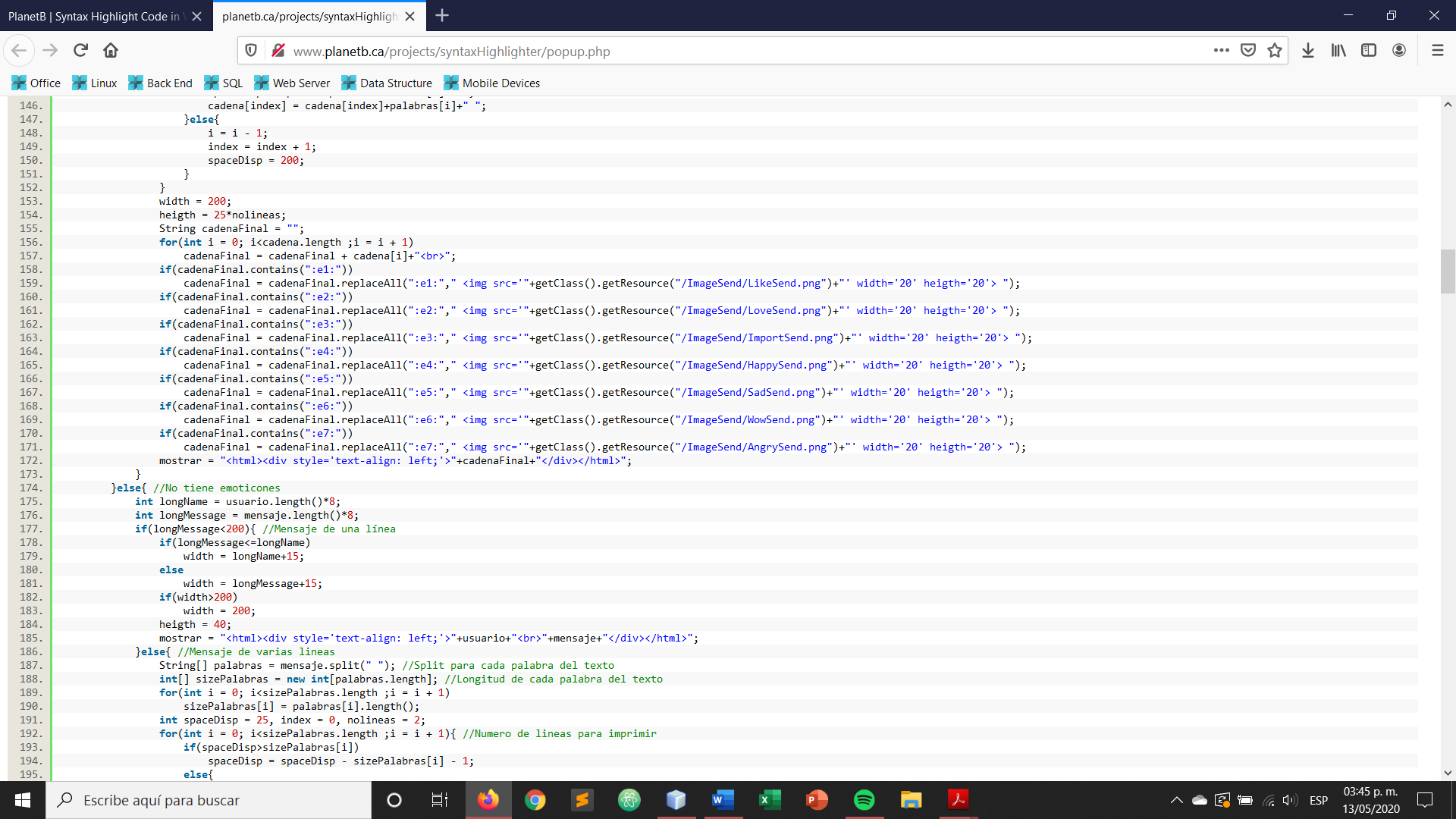
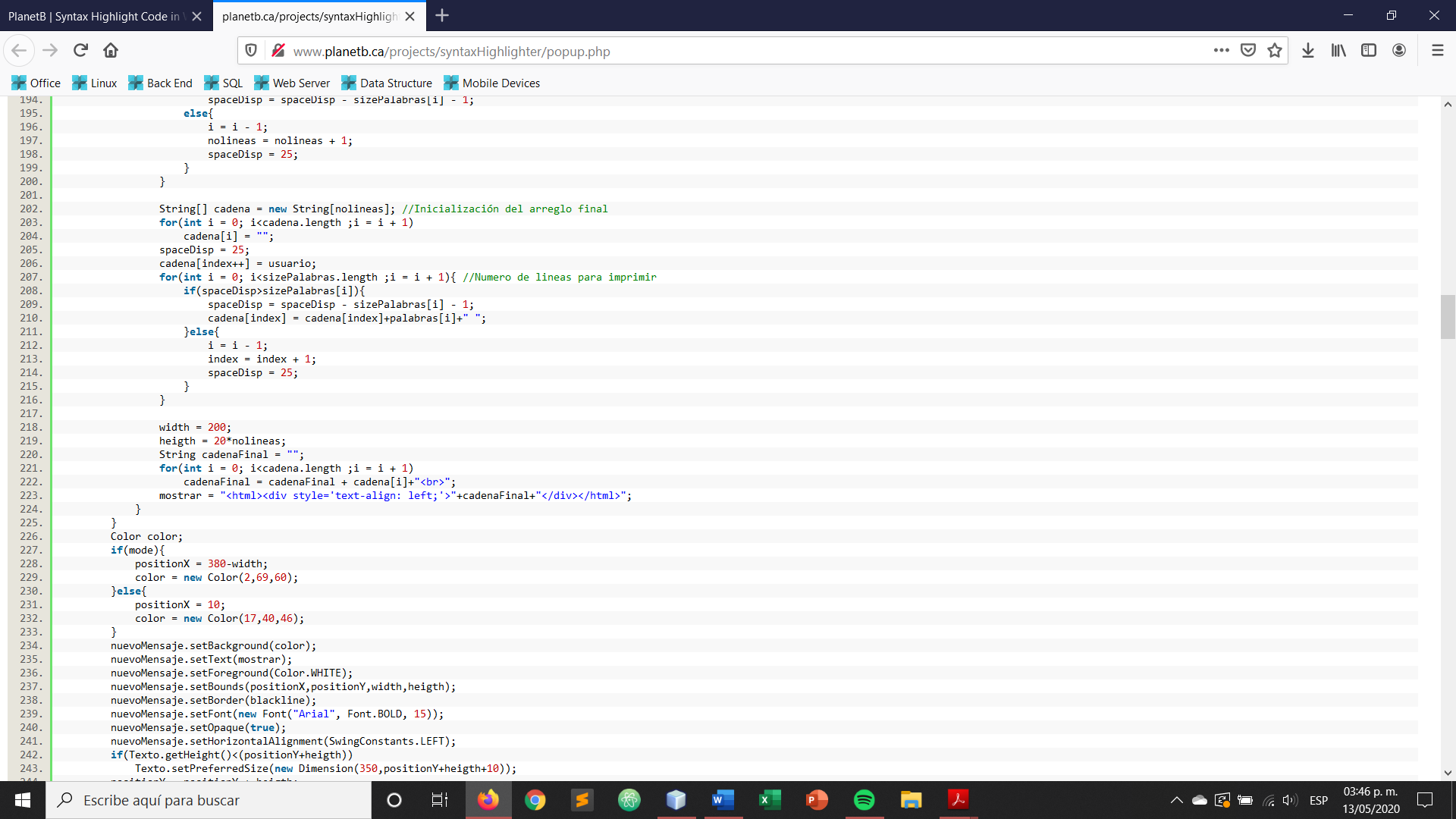
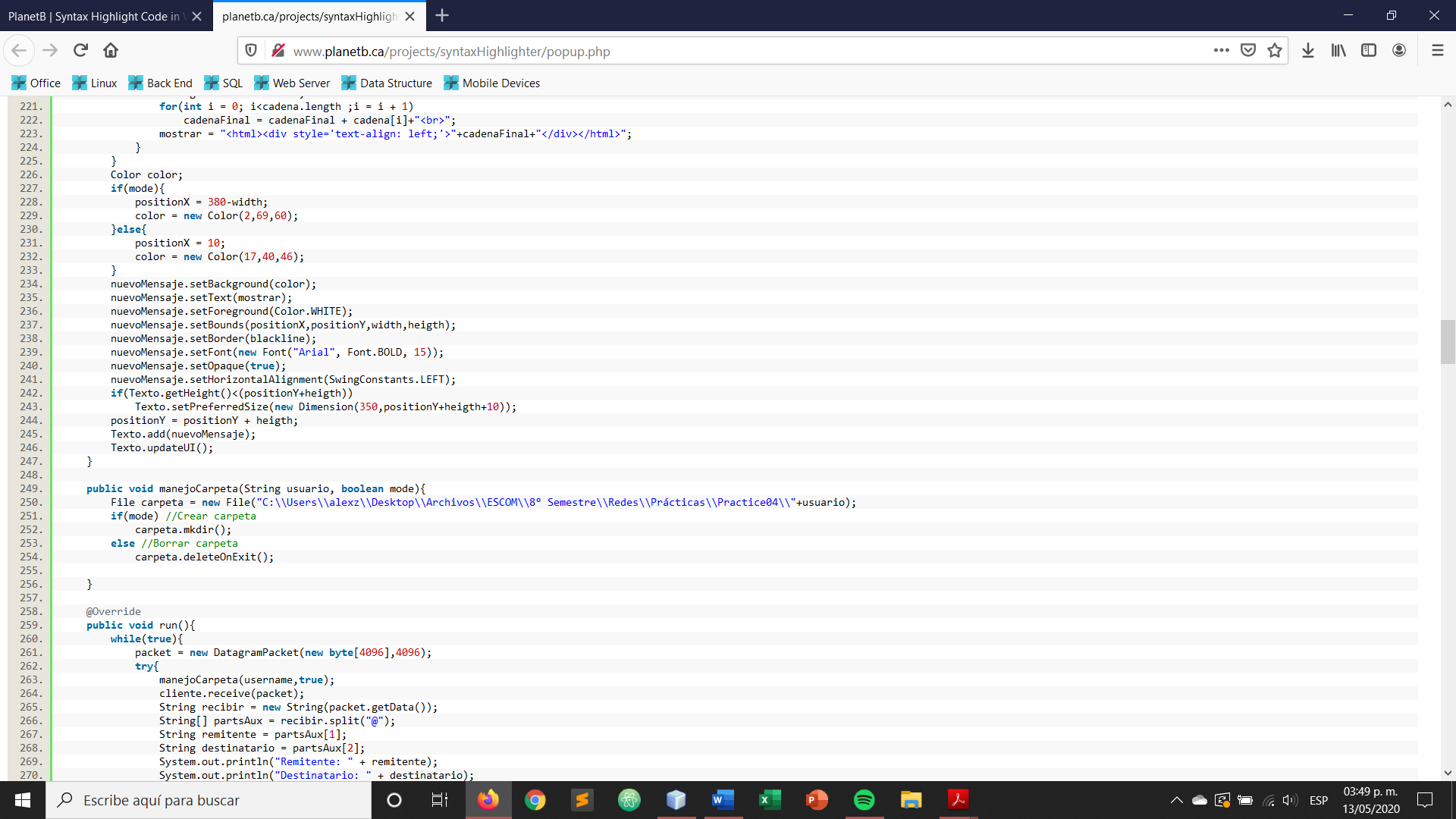
Realmente, todo el funcionamiento del servidor se encuentra en el main. Iniciamos con la creación del socket multicast y creamos el grupo de este. Después iniciamos un ciclo while-true para estar siempre disponible a peticiones por medio de Datagramas recibimos los mensajes del cliente, como el mensaje trae toda la información la desglosamos y empezamos el tratamiento del mensaje.

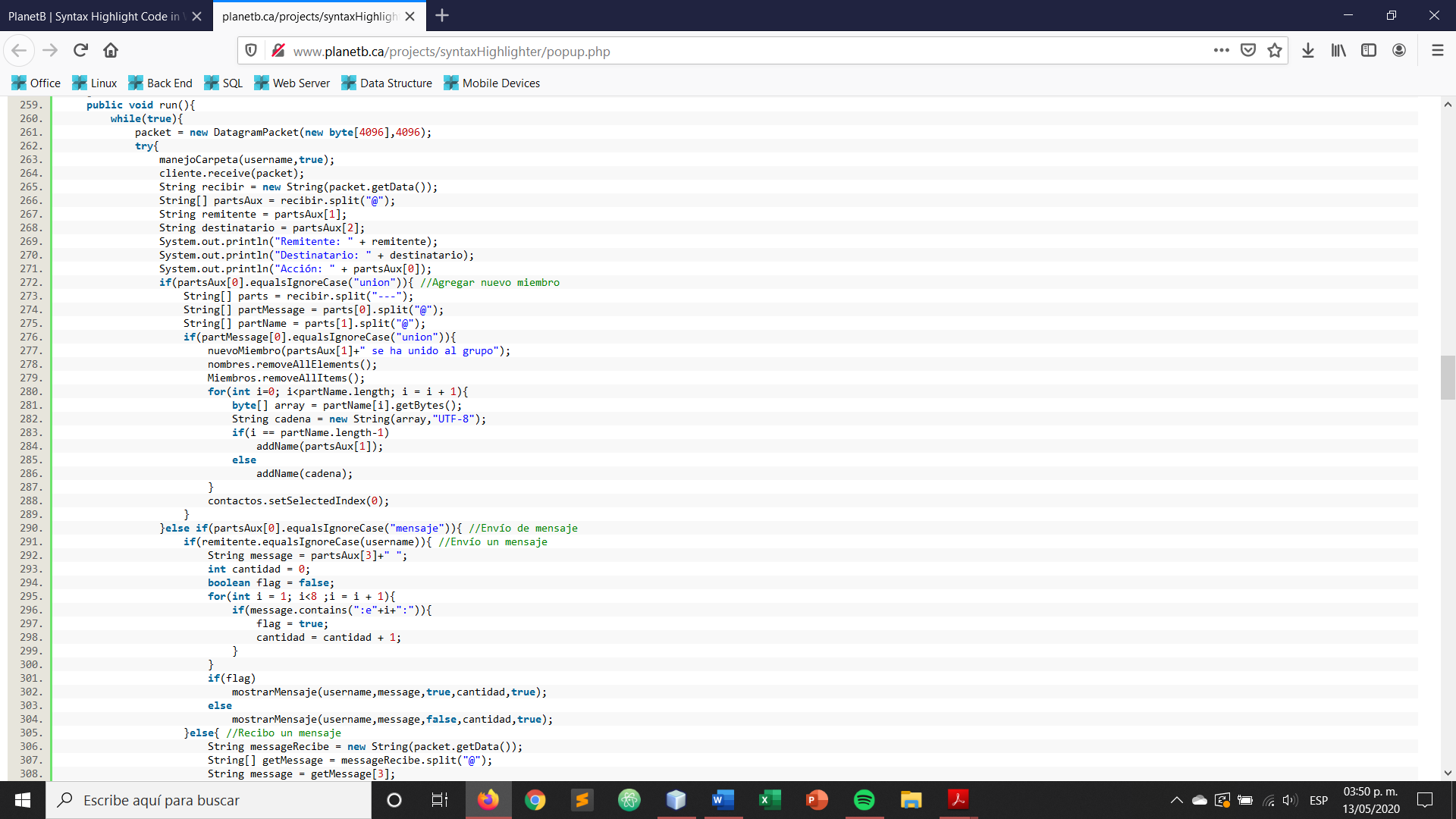
Ahora, vamos a explicar la parte del cliente.

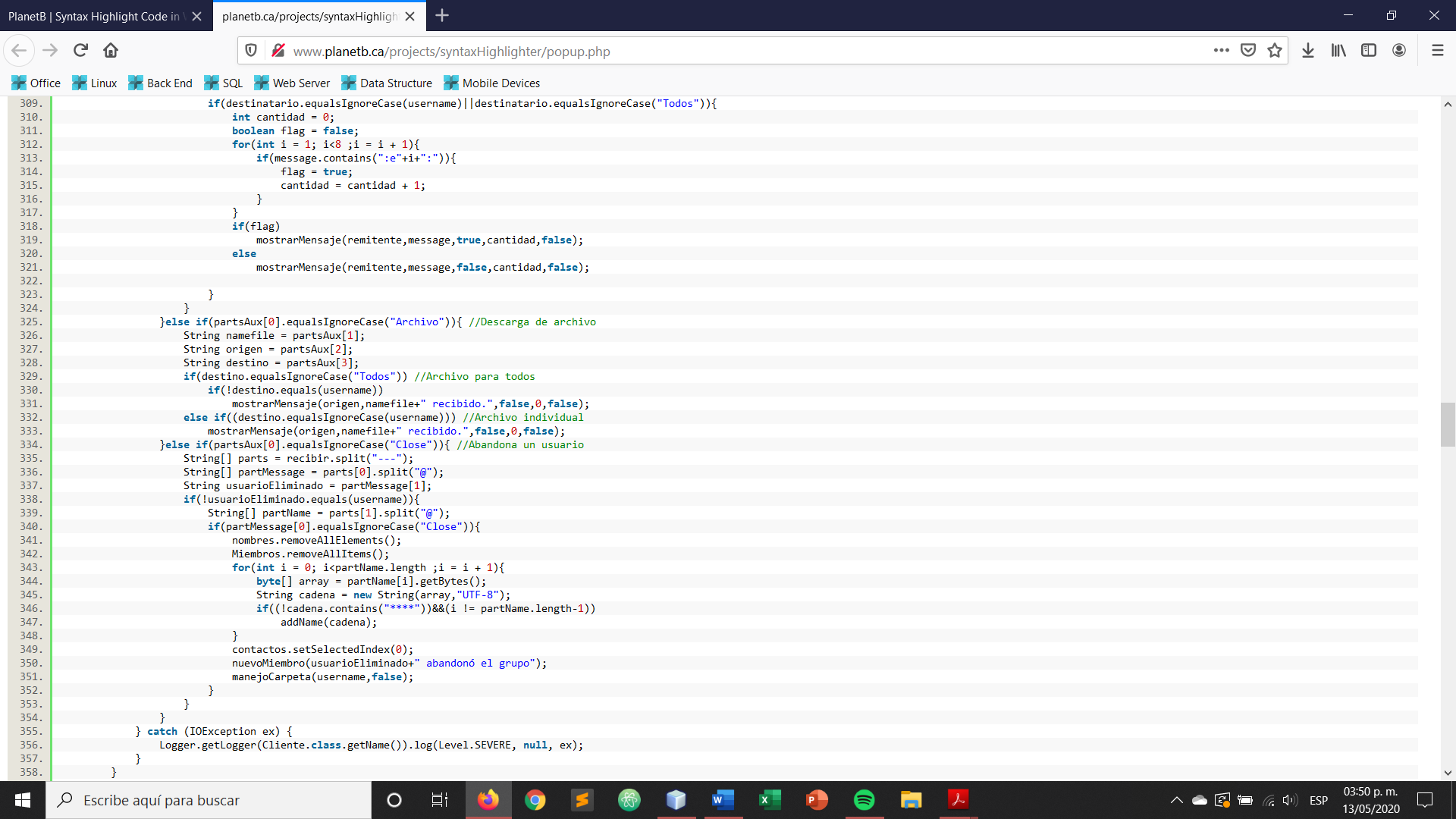
* **joinMe(String mensaje):** en este método iniciamos la conexión del cliente con el servidor, especificamos el funcionamiento del socket multicast y enviamos el datagrama con el mensaje para que lo pueda interpretar el servidor.
* **nuevoMiembro(String mensaje):** este método está dedicado para mostrar en la interfaz el mensaje de si un miembro se ha unido a la conversación o si alguien la ha abandonado. Además, para manejar el tamaño del panel, lo redimensiona si es necesario.
* **mostrarMensaje(String usuario, String mensaje, boolean emoji, boolean mode):** este método es el más importante para la interfaz gráfica, ya que es el encargado de desplegar el mensaje en la pantalla. Los dos primeros parámetros son obvios. Los dos siguientes son valores booleanos para saber como se debe tratar el mensaje, el primer valor indica si el mensaje tiene o no emoticones y el segundo para saber si es mensaje enviado o recibido.
* **run():** al igual que el servidor, corremos el cliente en un hilo para poder simular “Paralelismo” y controlar los cambios que sean individuales para cada aplicación y los que sean globales. El funcionamiento se centra en el mensaje recibido del datagrama, el primer parámetro indica la acción a realizar, el segundo el remitente y el tercero el destinatario. Posteriormente, con la acción se especifica lo que se tiene que hacer:
  + **Unión:** indica que un nuevo cliente se ha unido al grupo.
  + **Mensaje:** se trata de dos formas diferentes, si es enviado o recibido.
  + **Archivo:** maneja la lógica para el envío y descarga de archivos.
  + **Close:** indica que un miembro ha abandonado el grupo.

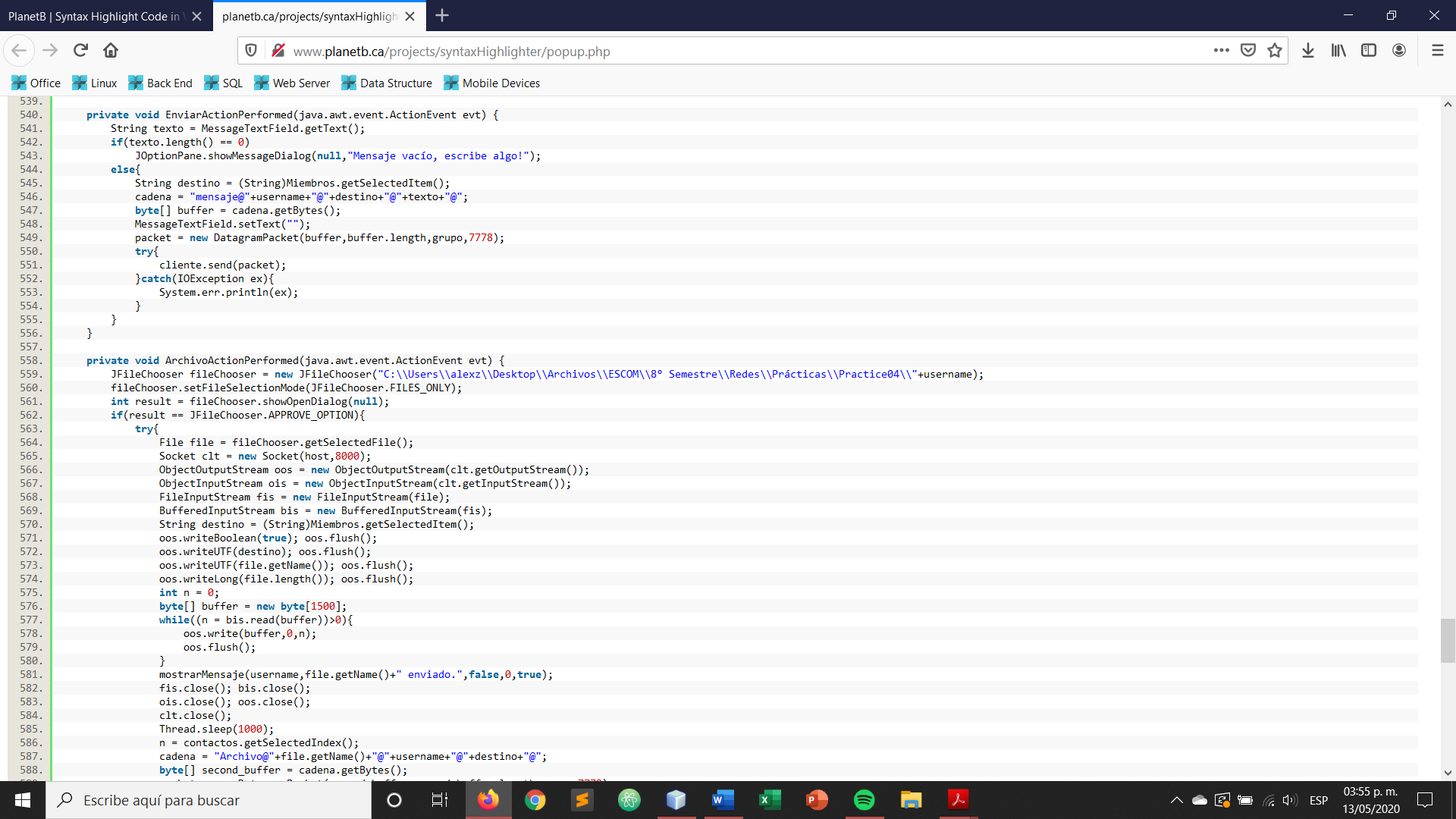
A continuación, se muestra el código del Cliente.

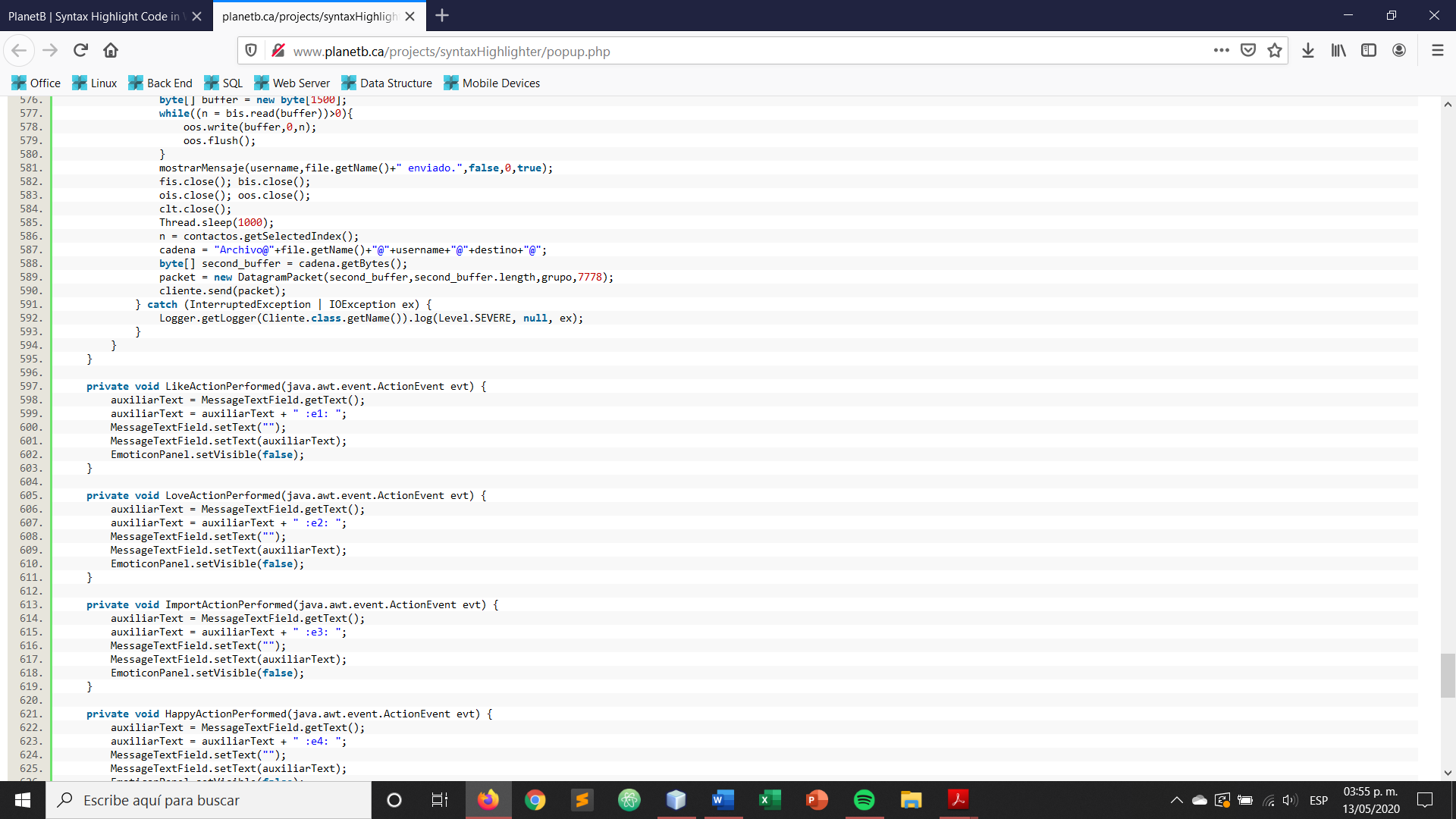




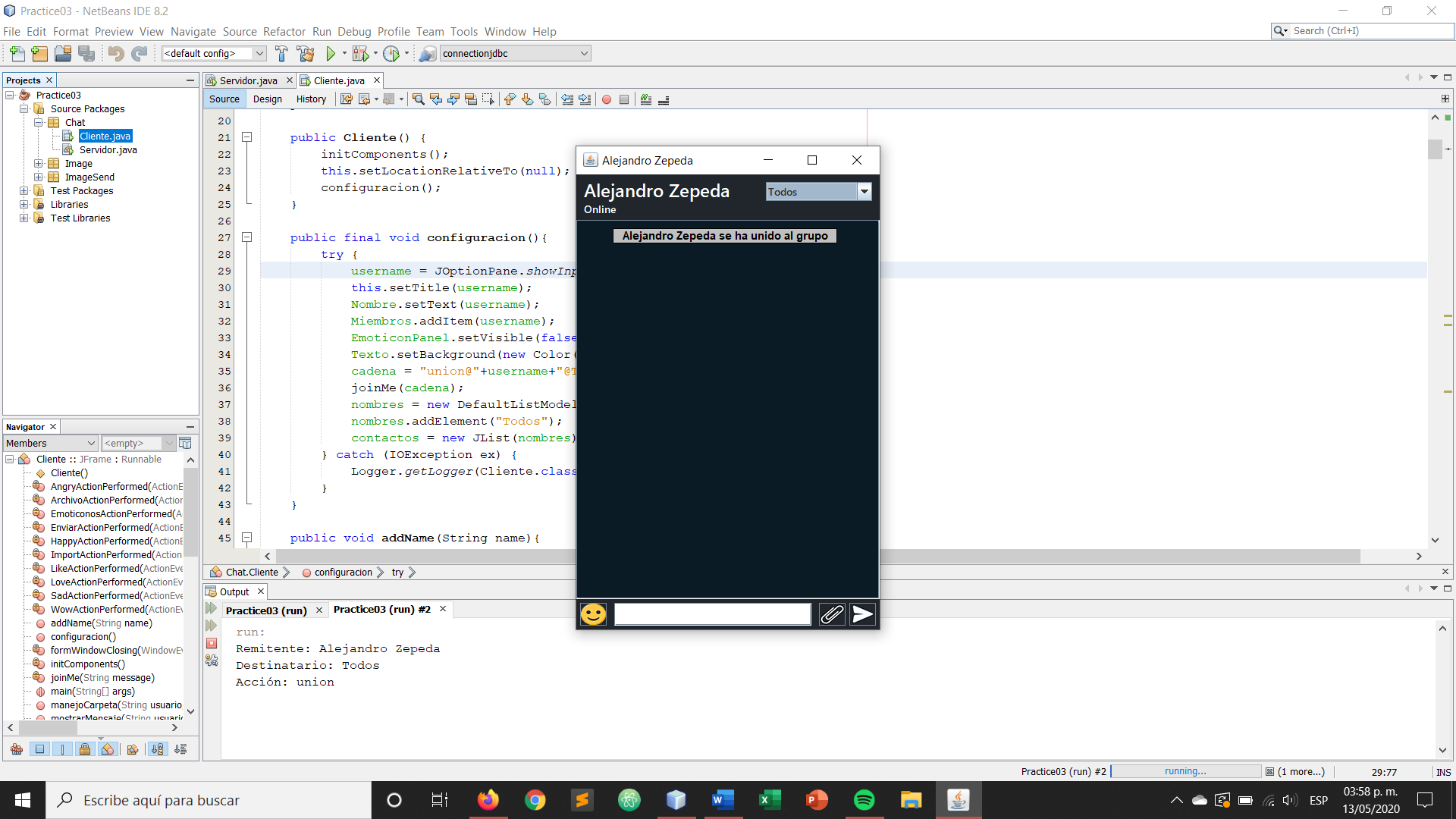






FUNCIONAMIENTO DE LA APLICACIÓN

A continuación, mostraremos un breve ejemplo de cómo funciona nuestra aplicación.

Esta es la pantalla principal, en el encabezado se muestra el nombre de usuario que previamente ingresaste. En la esquina superior derecha se observa un Combo Box donde se enlistan los nombres de todos los usuarios, se puede seleccionar un usuario individual o todos.

La parte central es en donde se irán mostrando los mensajes y notificaciones para el usuario.

En la parte inferior, se muestra un botón con un ícono de emoticono, al ser presionado se mostrará una lista de emoticones disponibles para enviar. Después aparece un Text Field para poder escribir el mensaje, seguido de un botón con un ícono de Clip, este se utiliza para poder mandar archivos. Por último, el botón de enviar.

Imagen 1. Interfaz general.

Para demostrar el funcionamiento, es necesario ingresar al menos 3 usuarios. Como se muestra en la imagen 2, a cada uno de los usuarios se les notifica si alguien se unió al grupo a partir de que ellos se unieron.



Imagen 2. Creación del grupo.

Como se observa en la imagen 3, aquí se puede apreciar el envío de un usuario a todos los destinos posibles. De igual manera, en el usuario central se enlistan las reacciones posibles que puedes agregar en un mensaje. Por último, en el usuario final se muestra el Combo Box con los usuarios disponibles para mandar mensaje.

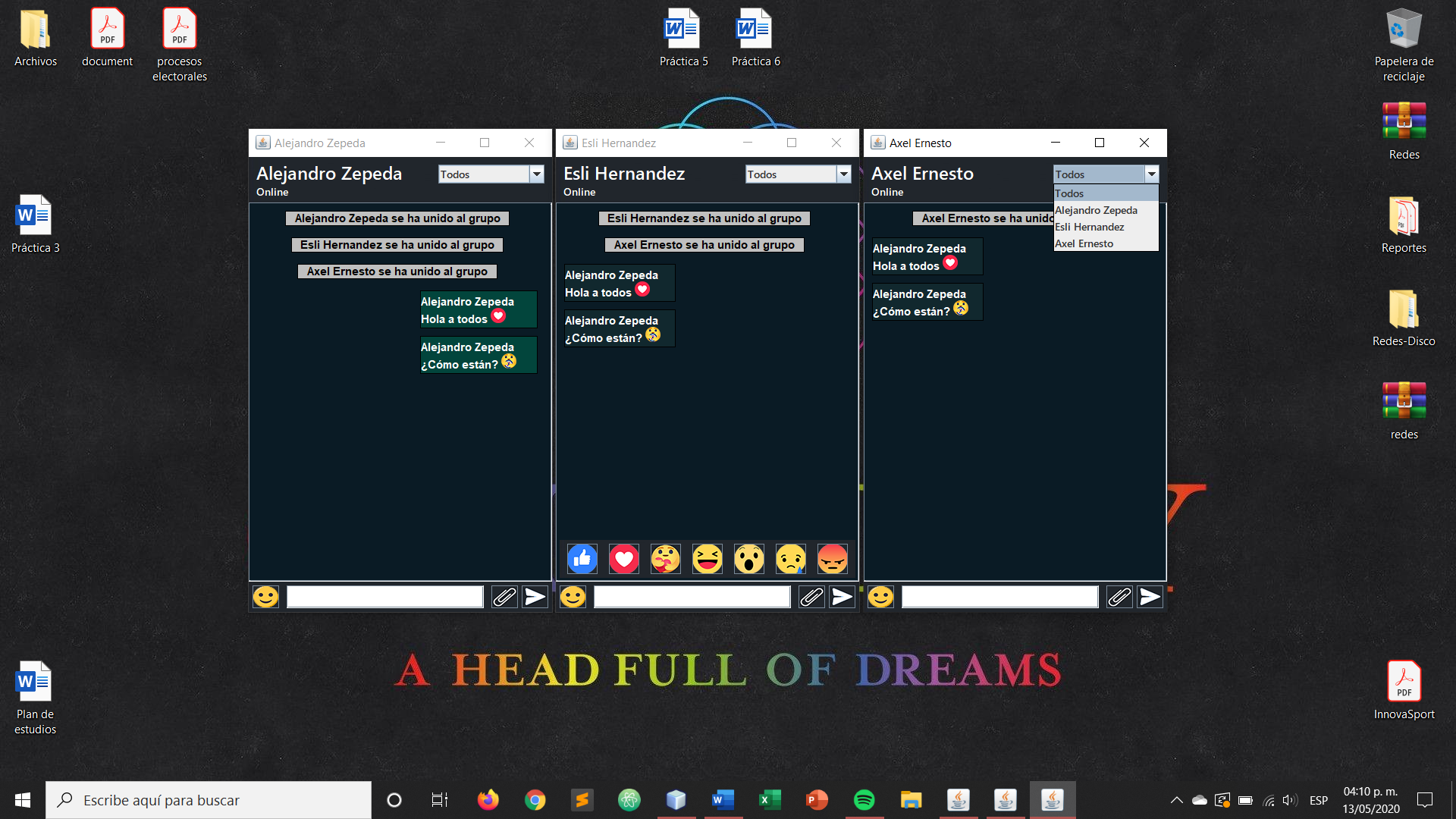


Imagen 3. Envío de mensajes con reacciones.

En la imagen 3, se muestra como el usuario inicial se comunica con el usuario central; existe un intercambio de mensajes, sin que el usuario final se entere de esta conversación. De igual manera, vemos como el usuario final se comunica con todos los usuarios.



Imagen 4. Envío de mensajes personalizados.

Dentro de la carpeta de la práctica, por cada cliente se crea una carpeta para simular el envío de archivos de manera personalizada. Vemos como el usuario inicial, se comunica con el central enviando un PDF y en la carpeta del usuario final, no existen modificaciones.

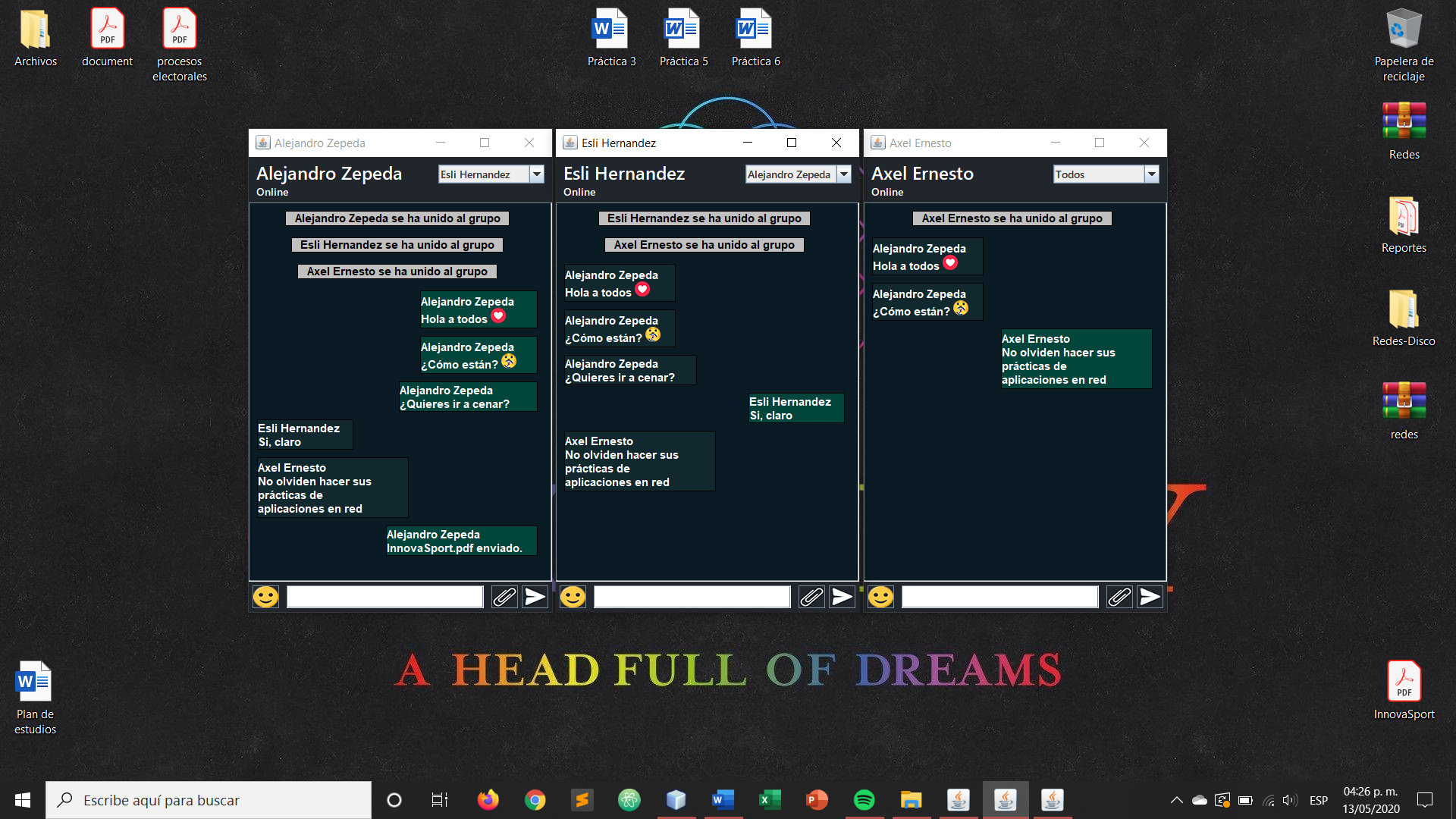
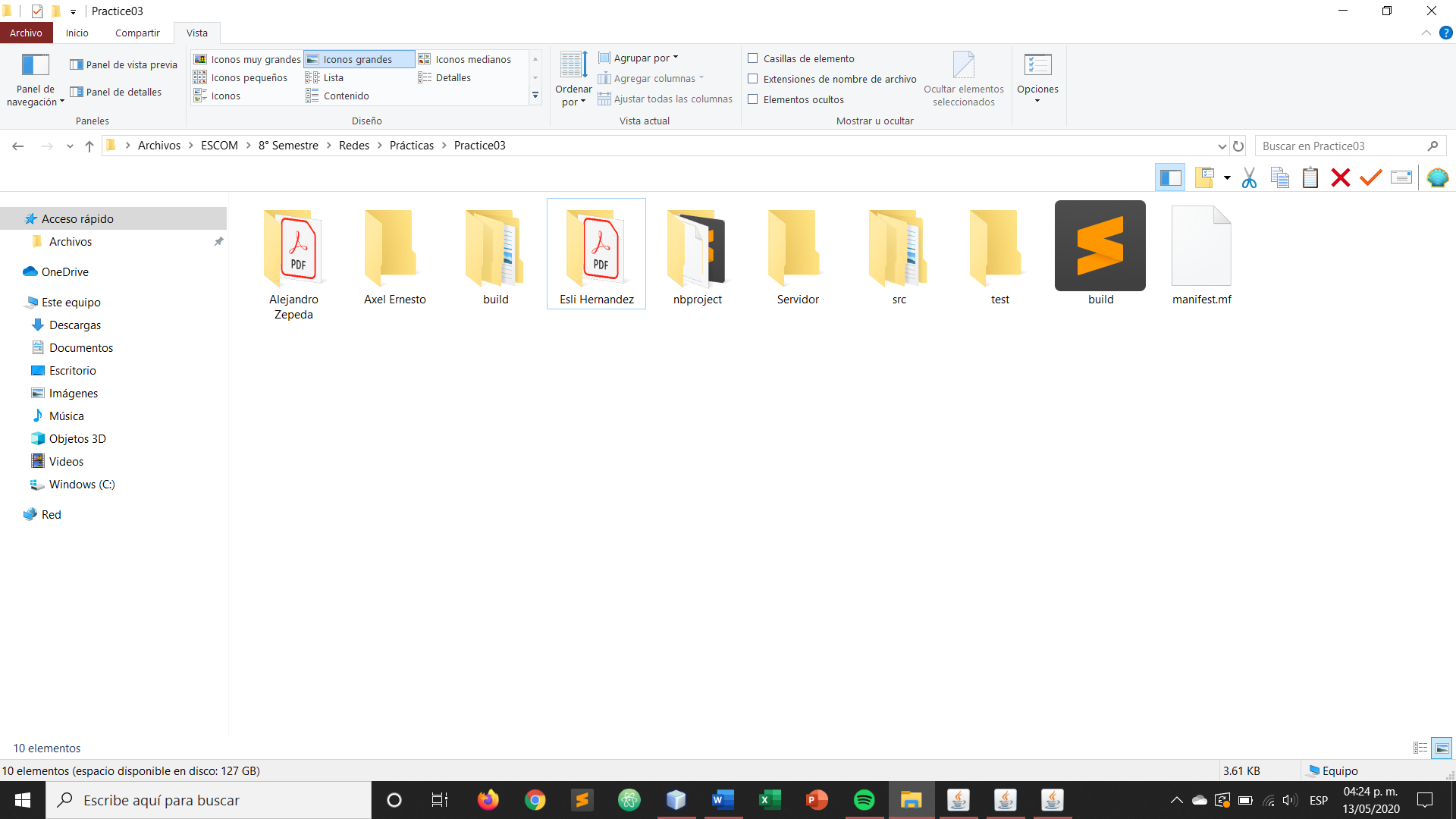


Imagen 5. Envío de archivos.

Por último, en la imagen 6, se muestra como el usuario final ha abandonado la conversación. El programa notifica al resto de los usuarios con un mensaje en la aplicación y de igual manera, en el Combo Box de los destinos posibles se actualiza para evitar confusión a la hora de enviar mensajes.

Además, en el usuario inicial, se muestra como el número de mensajes supero el tamaño inicial del panel, por lo que se redimensiona automáticamente y agrega un Scroll para poder visualizar los nuevos mensajes y no perder los anteriores.

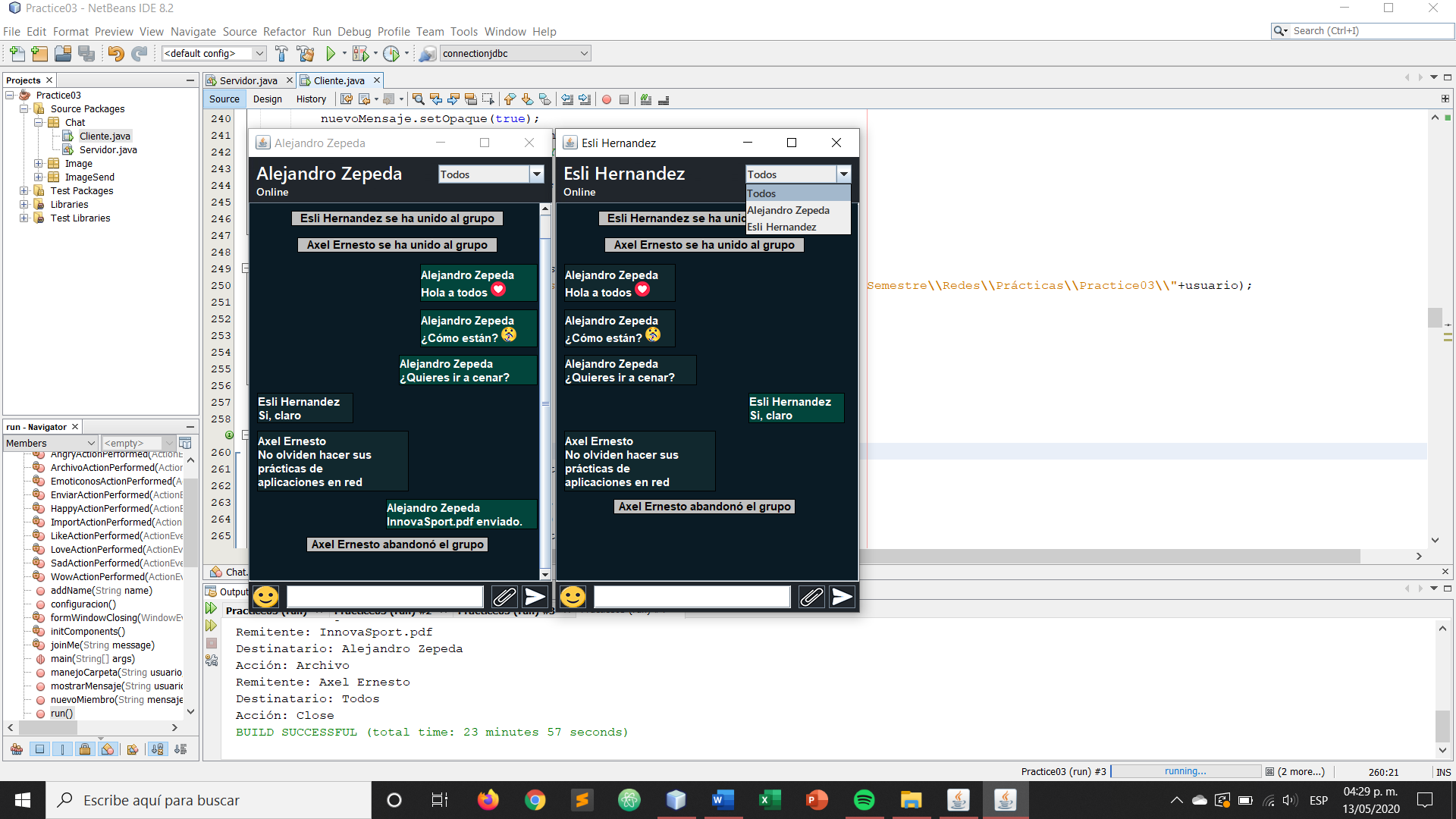


Imagen 6. Abandono de usuario.

CONCLUSIÓN

El uso de sockets multicast facilita la comunicación entre no solo cliente y servidor sino que también con otros clientes de manera específica, por lo que en esta práctica nos ayudó mucho para implementar grupos de chat y chat privados (dos personas) con operaciones básicas (envío de archivos y mensajes, reutilizando el conocimiento adquirido en las practicas anteriores), también se aprendió como se manejan los emojis en los sistemas con su respectivo código ünicode" que es el que utilizamos en la aplicación de chat.