INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

REDES DE COMPUTADORAS

PROF. AXEL ERNESTO MORENO CERVANTES

GRUPO: 3CM7

PRÁCTICA 6 – CALCULADORA RMI

ALEJANDRO DE JESÚS ZEPEDA FLORES

21 de junio de 2020

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

Práctica 6: https://drive.google.com/drive/folders/1Tc4wEb7ZKgaCSFX-SHNAYRjMgjLCNc6m

OBJETIVOS

Implementar una calculadora distribuida, utilizando al menos dos servidores remotos (Java RMI), el primer servidor contendrá las operaciones aritméticas, mientras que el segundo servidor tendrá la implementación de operaciones trigonométricas.

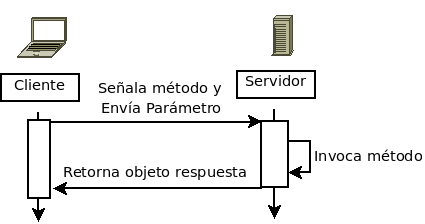
INTRODUCCIÓN

**RMI: Remote Method Invocation**

RMI es una tecnología desarrollada por Sun para permitir la colaboración de objetos que están localizados remotamente. Esta tecnología se enmarca en la idea de permitir colaboración entre Objetos Remotos. La idea no es que los objetos se comuniquen a través de la programación del usuario de protocolos estándares de red.

La idea es tener un objeto cliente, donde podamos completar un requerimiento de datos. El cliente luego prepara el requerimiento que envía a un objeto ubicado en un servidor. El objeto remoto prepara la información requerida (accediendo a bases de datos, otros objetos). Finalmente, el objeto remoto envía la respuesta al cliente. En lo posible esta interacción debería ser lo más semejante posible a requerimientos hechos localmente.

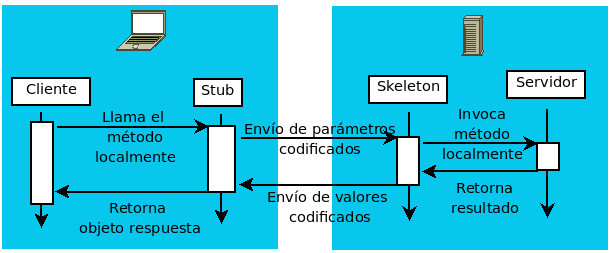
En principio se puede anhelar la colaboración de objetos escritos en cualquier lenguaje (no es el caso de RMI). Esta idea no es simple de lograr, corresponde al esfuerzo del grupo OMG los cuales propusieron CORBA (Common Object Request Broker Architecture), el cual define un mecanismo común para descubrir servicios e intercambiar datos. CORBA usa Object Request Broker (ORB) como traductores universales para la comunicación entre objetos. Los objetos remotos hablan a través de estos ORB. El protocolo de comunicación entre objetos y ORB es llamado Internet Inter-ORB Protocol o IIOP.



* **Objeto cliente**: objeto cuyo método hace el llamado remoto.
* **Objeto servidor**: Objeto remoto llamado

Notar que los roles de cliente y servidor aplican sólo a un llamado. Un objeto servidor luego puede ser cliente al hacer un llamado remoto.

* **Marshalling**: es el proceso de codificación de los parámetros.
* **Stub**: es un objeto que encapsula el método que deseamos invocar remotamente. Así el llamado remoto es semejante a un llamado local. Éste prepara información con la identificación el objeto remoto a invocar, el método a invocar y codificación de los parámetros (Marshalling).
* **Skeleton**: es el objeto del lado del servidor que decodifica los parámetros, ubica el objeto llamado, llama el método deseado, codifica el valor retornado, y envía la información de regreso al stub.



Aun cuando el proceso de la Figura 2 es completo, RMI lo hace en gran medida automático y en gran media transparente para el programador. La sintaxis de llamados remotos es la misma de los llamados locales.

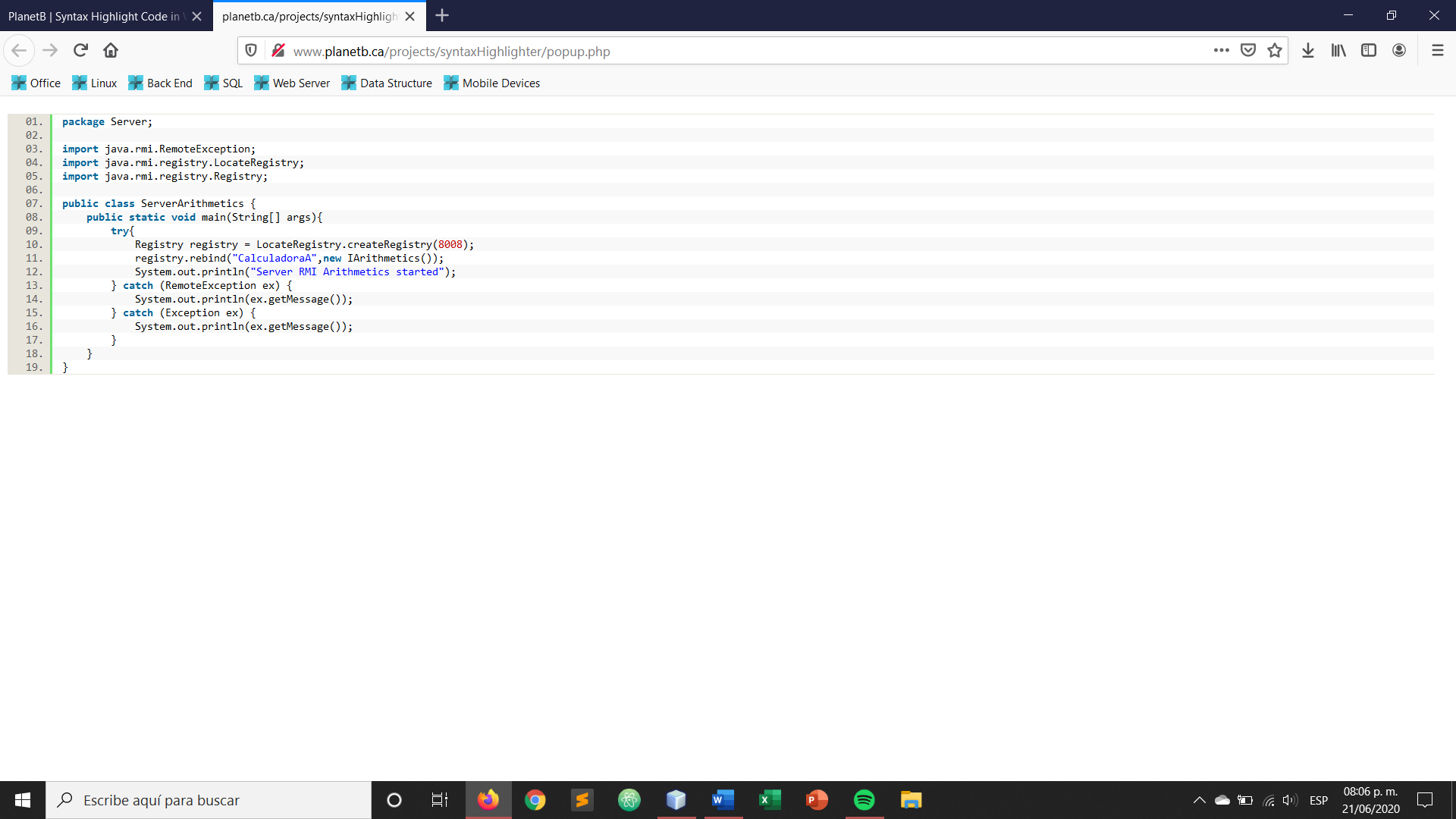
RMI posee un mecanismo para cargar clases dinámicamente desde otro lugar. Esto es requerido, por ejemplo, cuando el valor retornado corresponde a una instancia de una clase derivada de la clase conocida por el cliente. Aquí se ocupa un mecanismo similar al usado por applets.

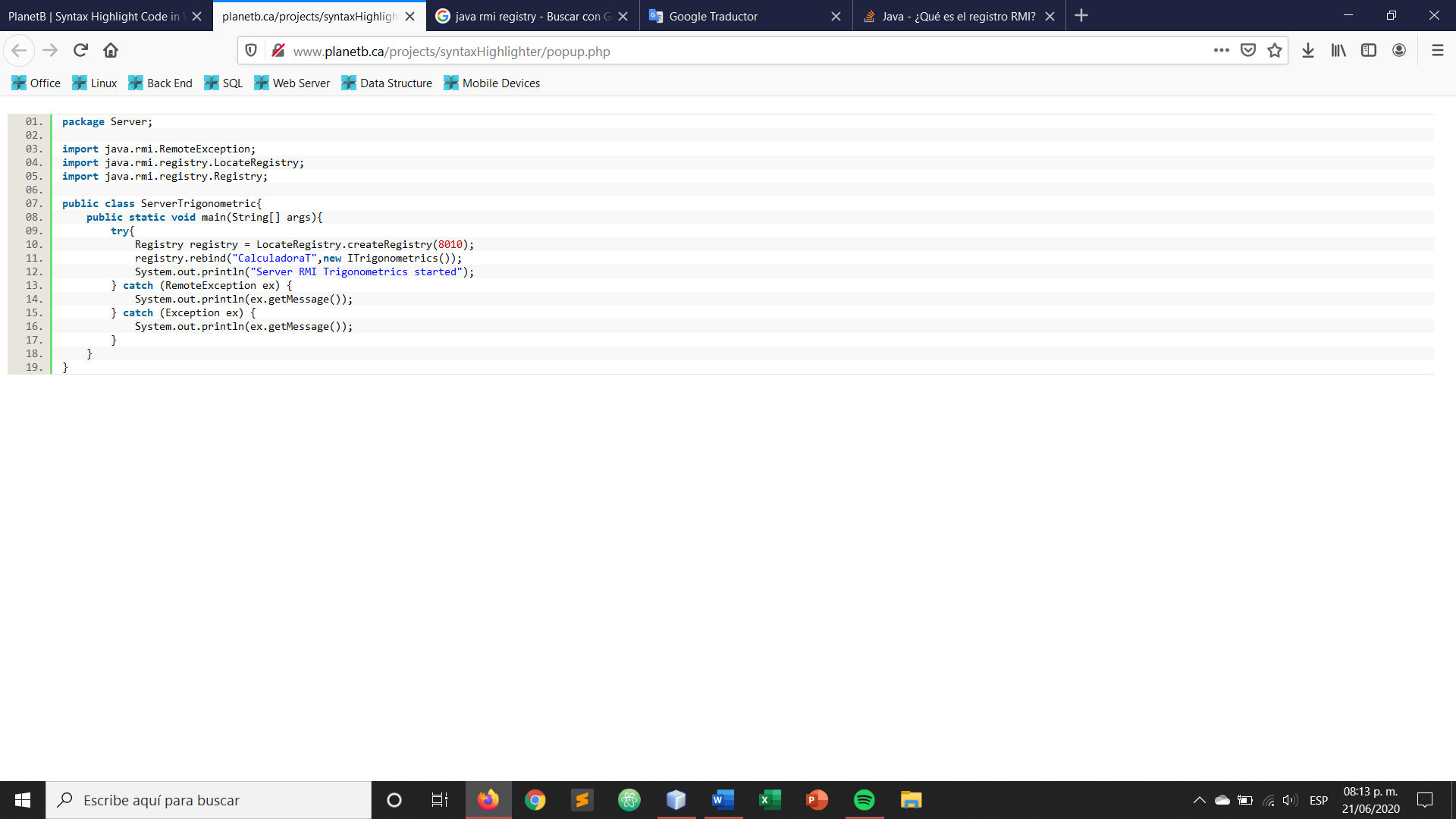
**¿Por qué utilizar RMI y no RCP?**

La principal diferencia de utilizar RPC o RMI, es que RMI es un mecanismo para invocación remota de procedimientos basado en el lenguaje de programación Java que soporta interacción entre objetos, mientras que RPC no soporta esta característica.

DESARROLLO

La implementación de esta práctica fue un poco robusta, debido a que se tuvieron que codificar dos servidores con sus respectivas clases remotas. Empezamos con la parte del Servidor de operaciones aritméticas y trigonométricas.



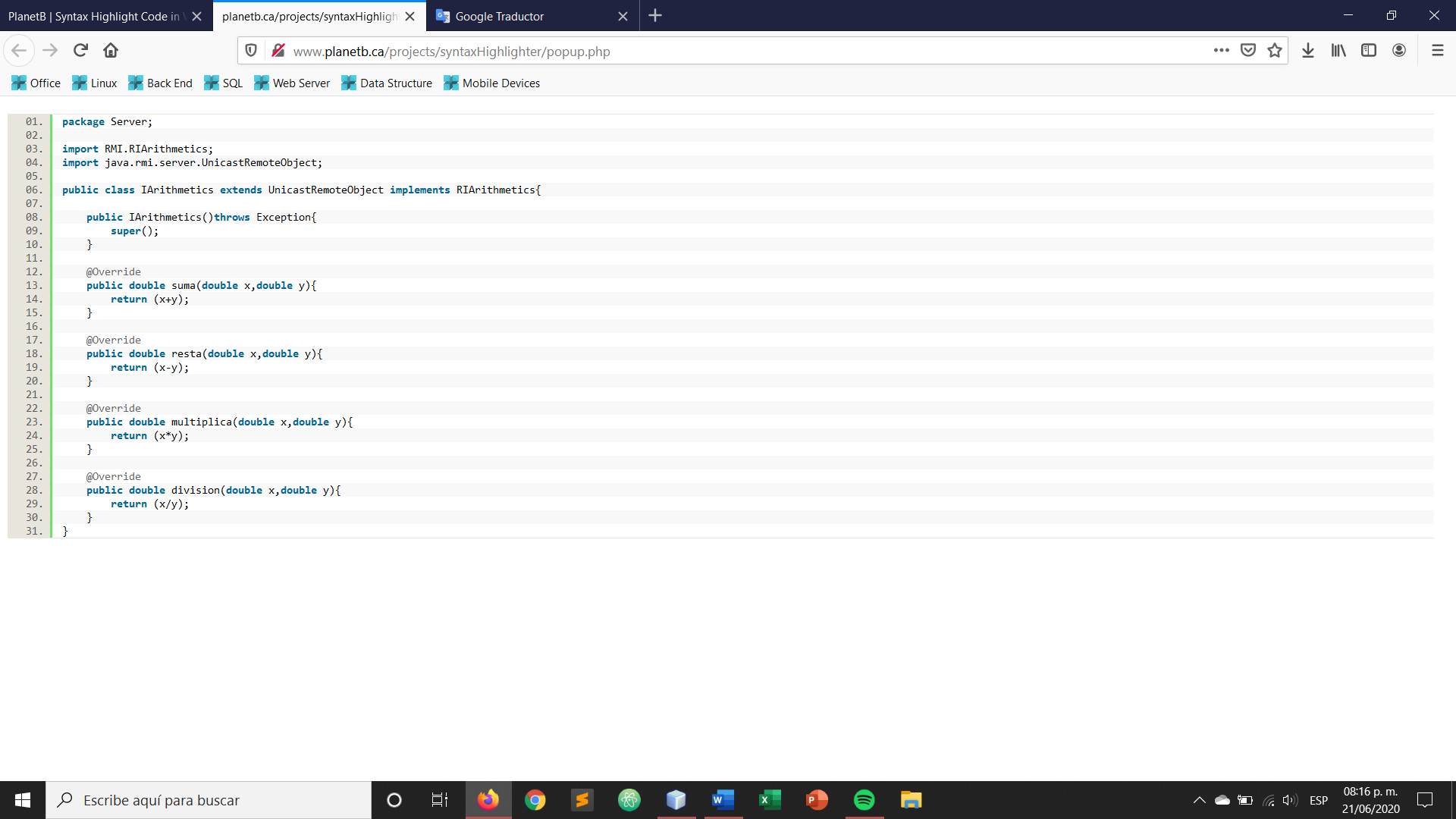


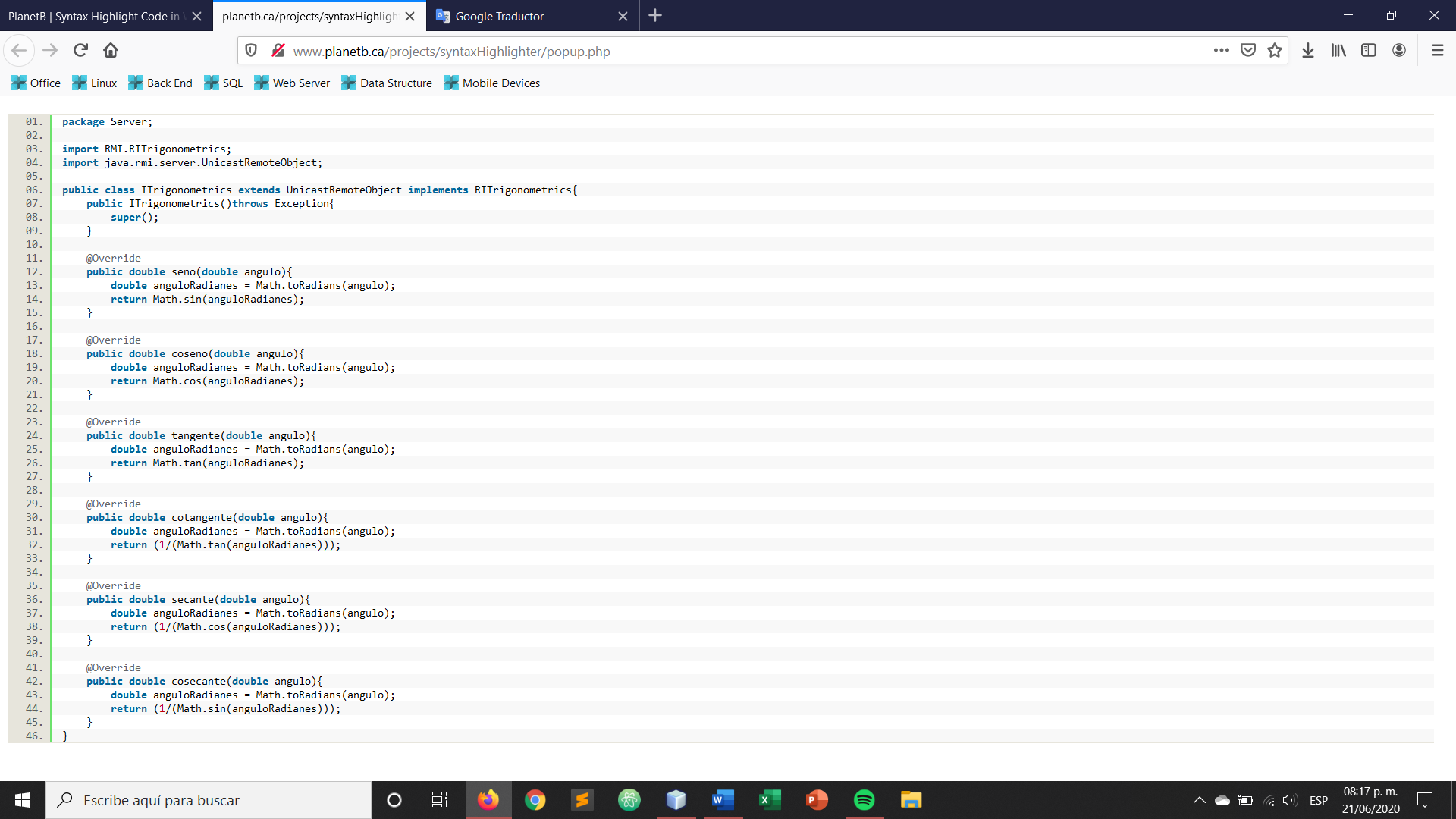
A diferencia de los servidores implementados en prácticas pasadas, notará que su codificación no es muy extensa. Únicamente, dentro del método main, debemos utilizar la interfaz “Registry”. Es un componente clave del sistema RMI que proporciona un directorio centralizado para que los servidores registren servicios y para que los clientes busquen estos servicios.

Después, con el método rebind, registramos los métodos que ofrecerá el Servidor. Además, las únicas diferencias entre ambos servidores, es el puerto en el que registramos el servidor y los métodos que implementan cada uno.

A continuación, explicaremos las clases con los métodos implementados.

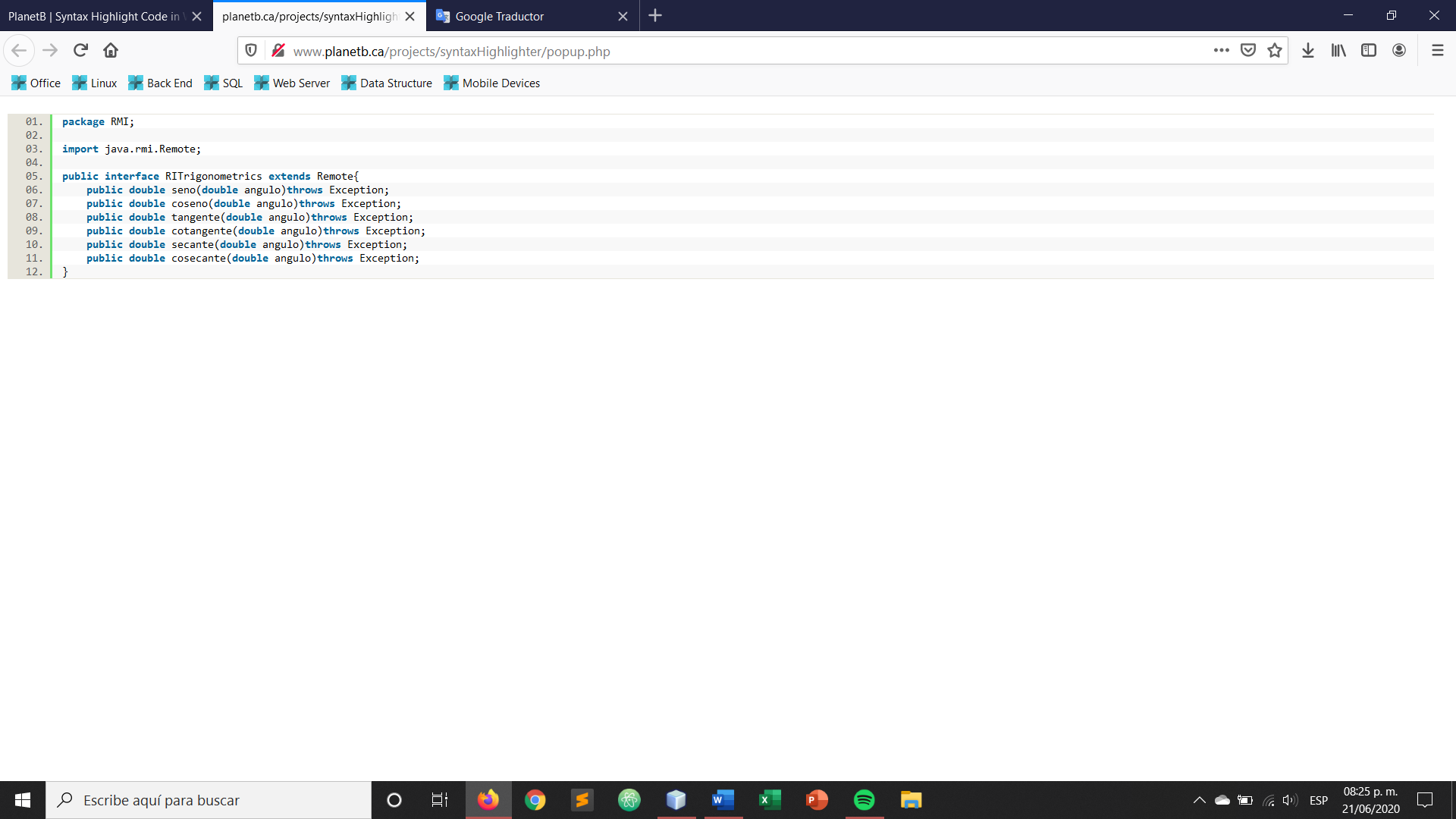
Ambos métodos extienden la clase UnicastRemoteObject, sin embargo, implementan diferentes interfaces. Estas interfaces, que explicaremos posteriormente, tienen únicamente la definición de los métodos, mientras que las clases tienen la implementación de estos.

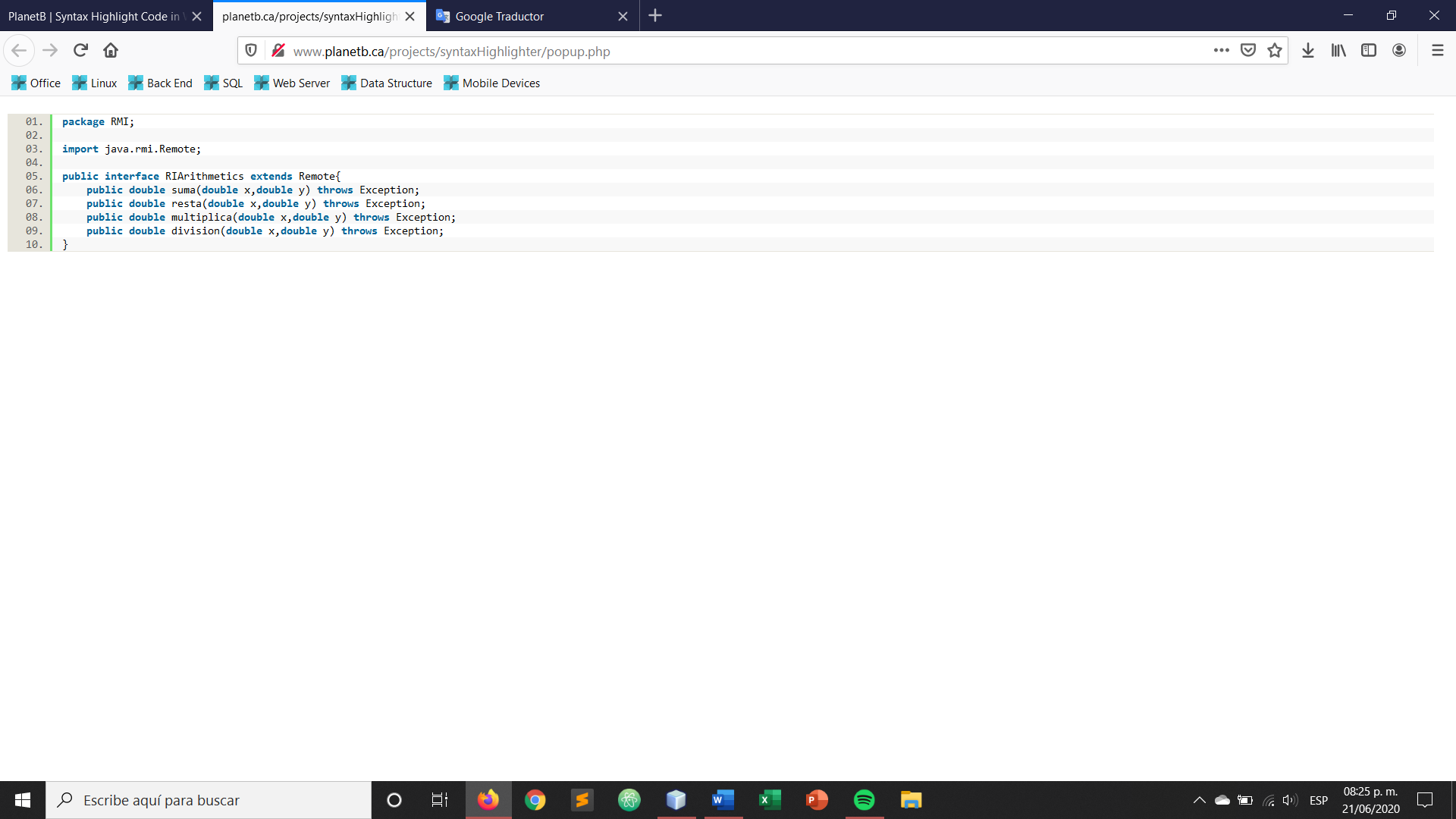




Terminamos, con la parte del servidor. A continuación, explicaremos las interfaces que, igualmente extienden de la clase Remote. Estas interfaces estarás disponibles para el usuario localmente y son las que el cliente utilizará para llamar al método localmente, mientras que las interfaces se comunicarán con el Servidor.

Como se puede observar, el código de ambas interfaces es la definición de los métodos.

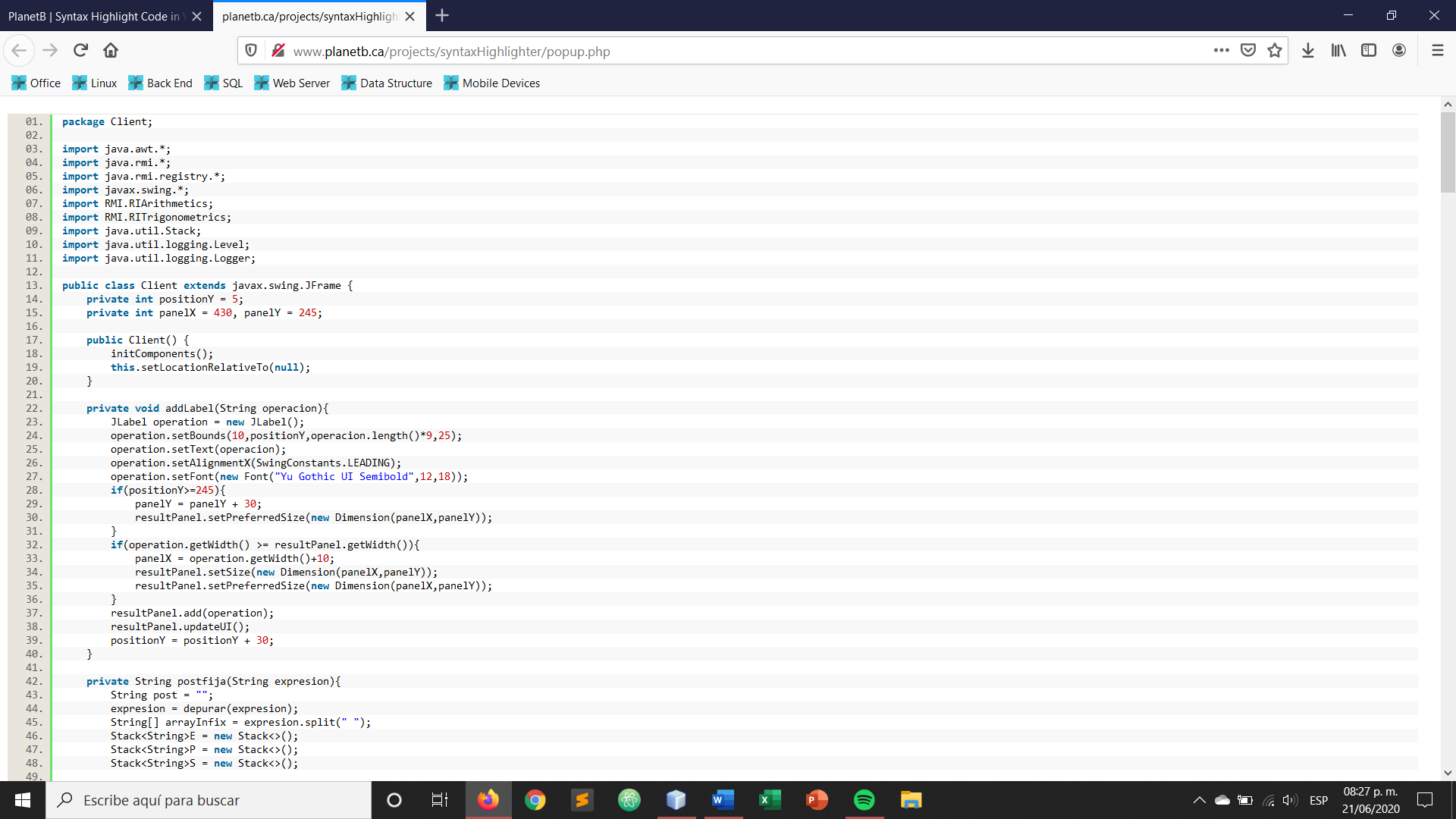


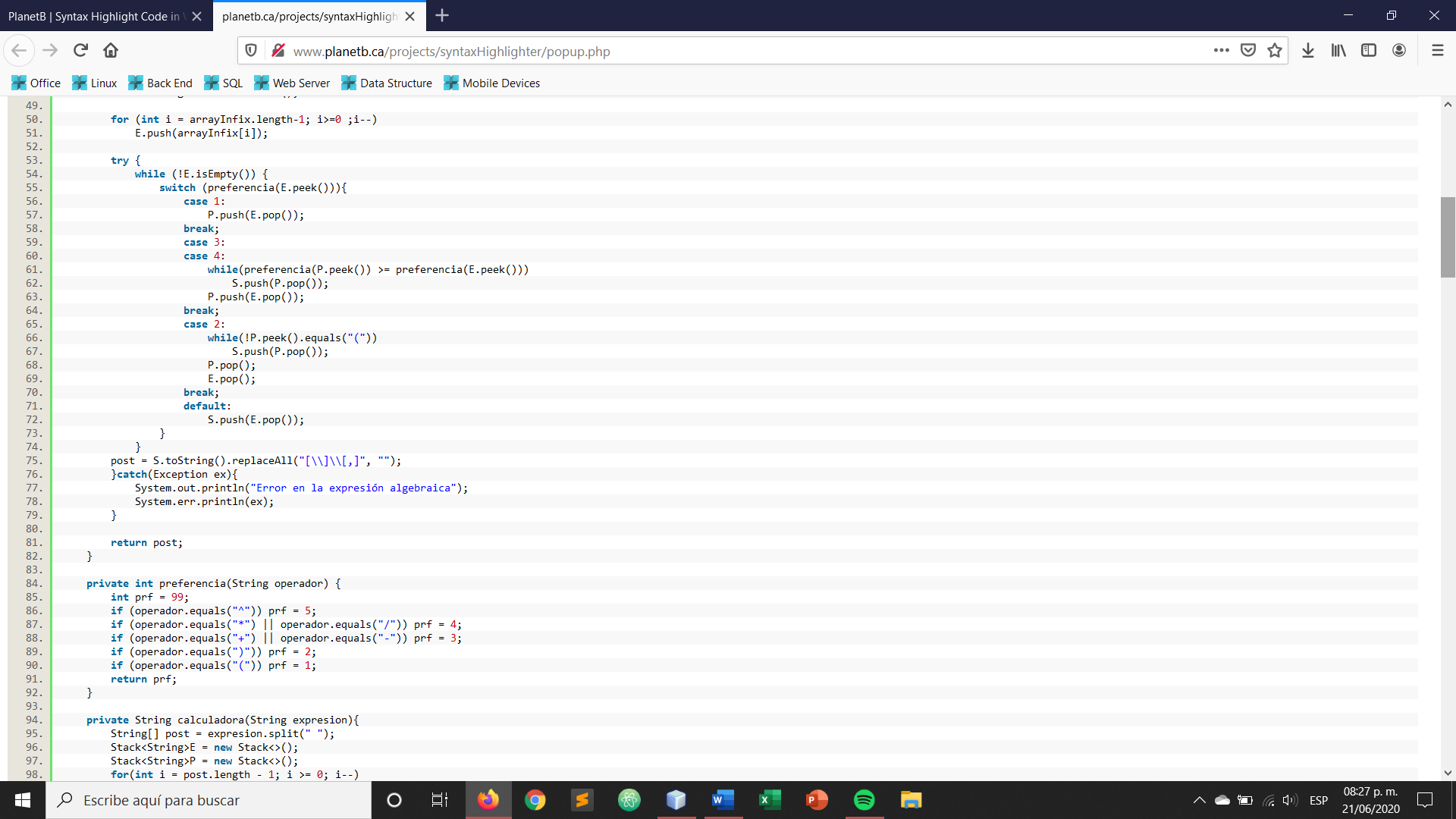


Por último, terminamos con la parte del cliente.

Como el cliente es la única clase que está implementada orientada a la interfaz gráfica, explicaremos estos métodos primero. El primero es **addLabel(String operación)**, este método crea un JLabel con el contenido especificado en el parámetro de la función e imprime en el JPanel la operación realizada por el usuario y el resultado de esta.

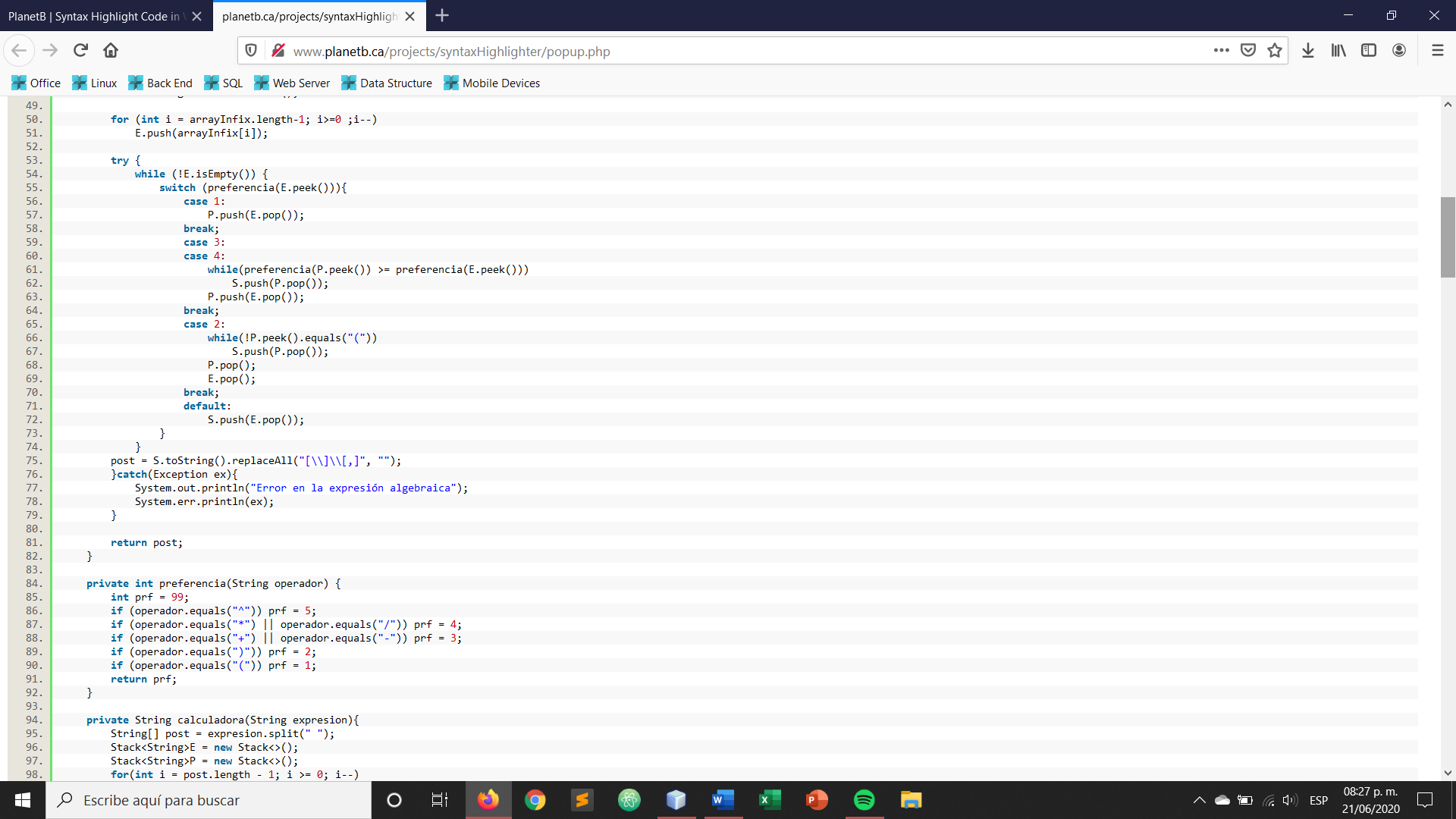
Después, se encuentra el método **posfijo(String operación)**, este método se encarga de convertir la ecuación de infijo a posfijo. Este método fue utilizado en estructuras de datos y resulta una manera muy sencilla de evaluar expresiones.

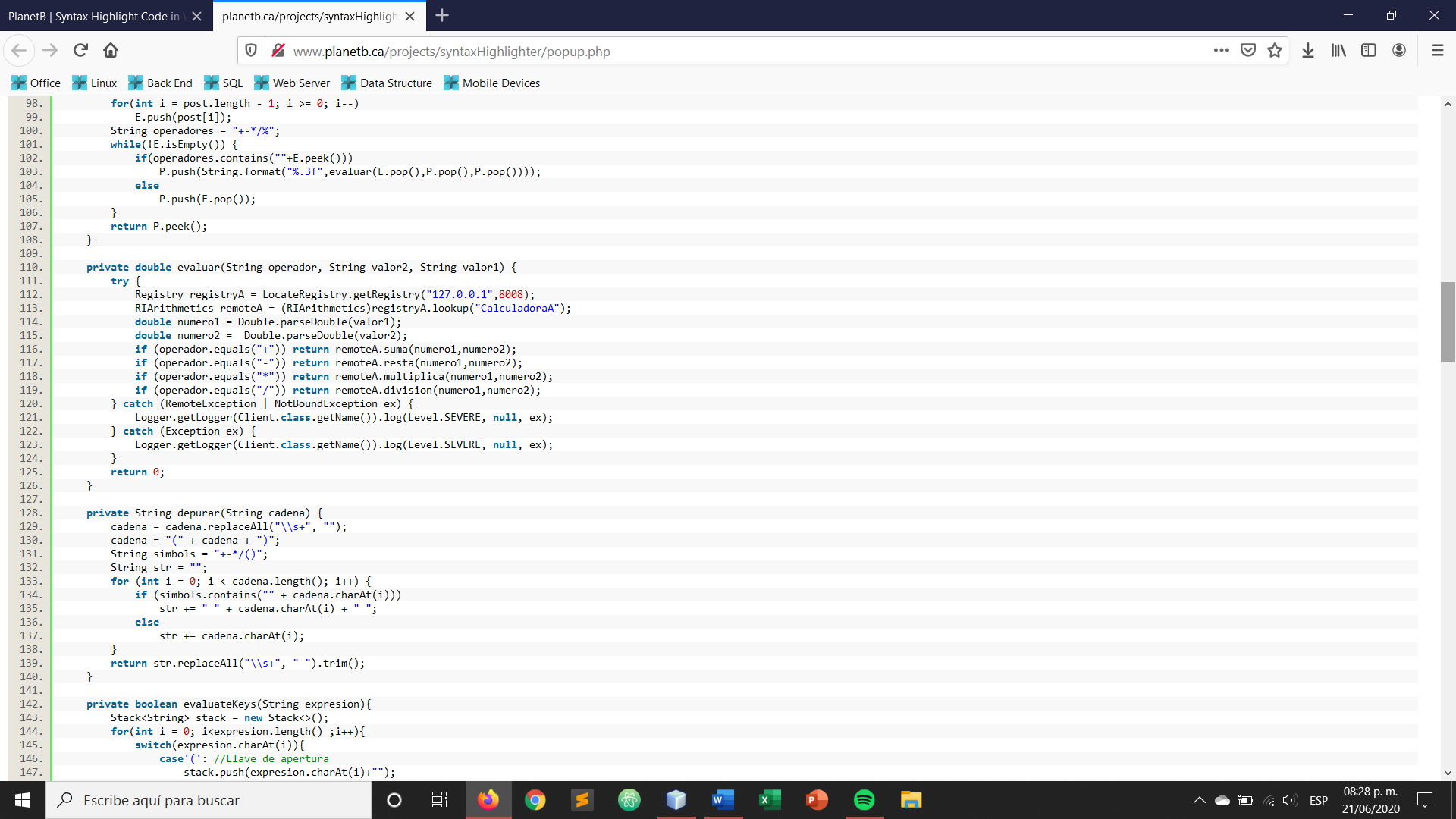




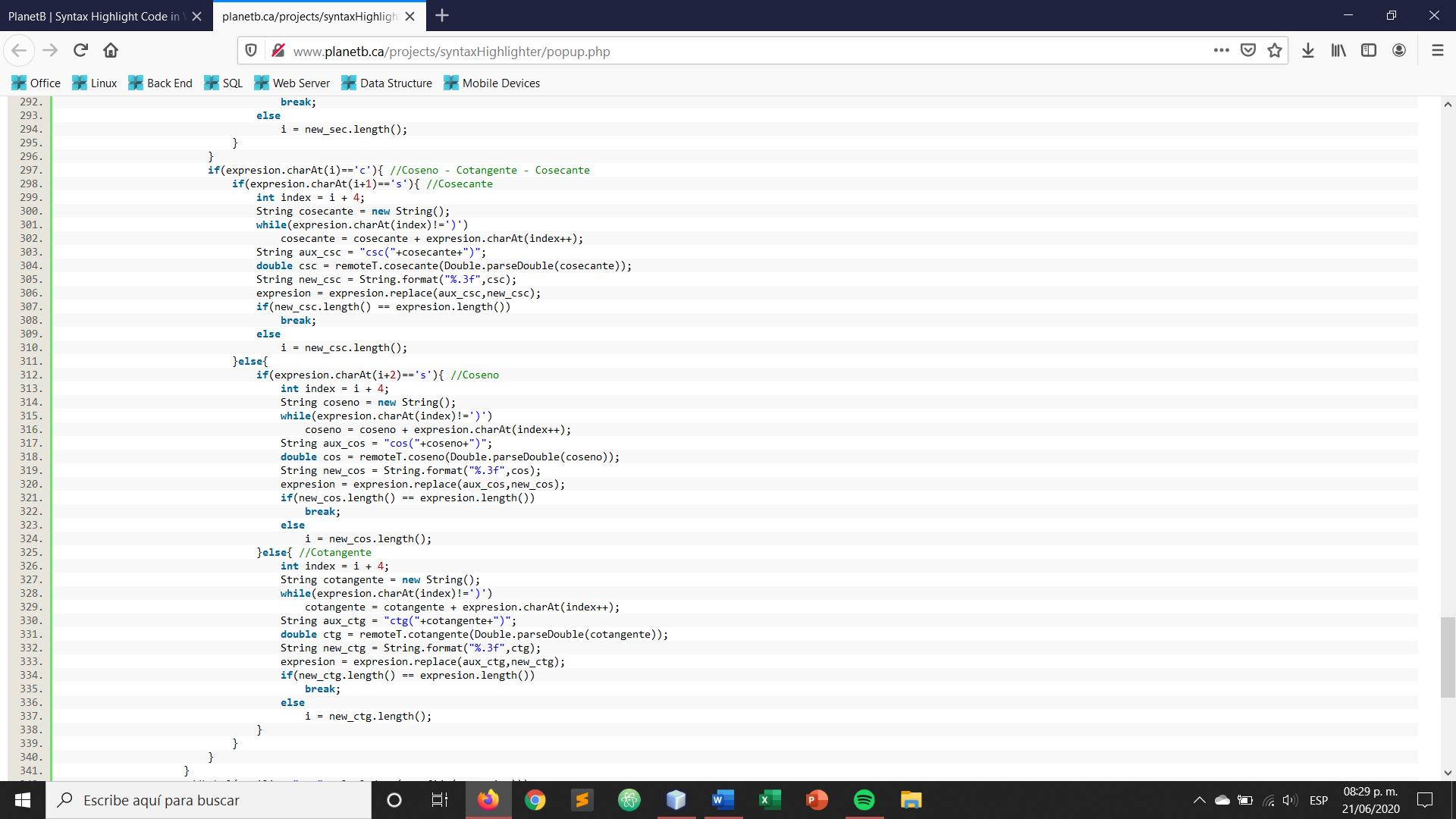
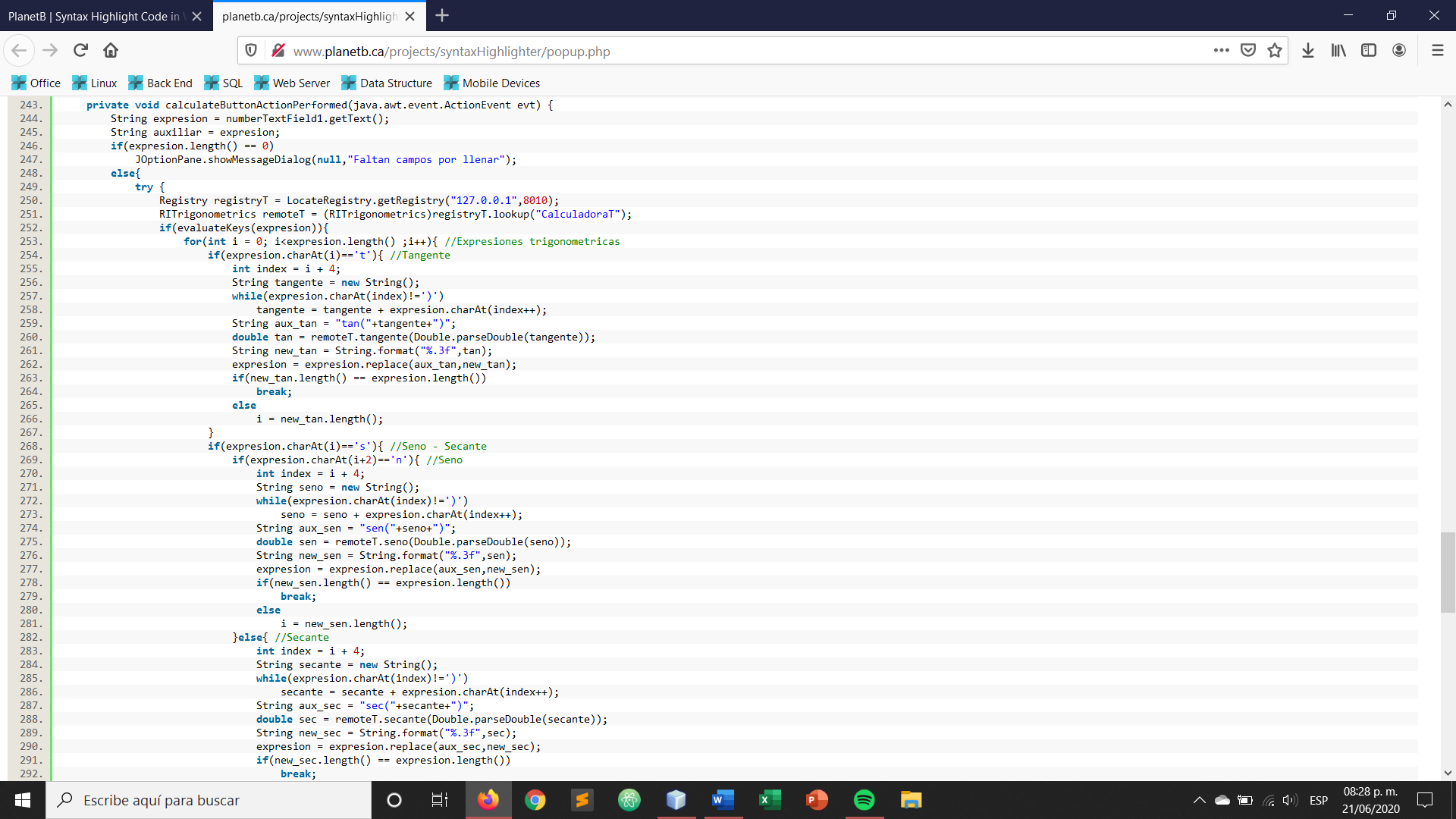
A continuación, se explicarán los métodos más importantes.

El método **calculadora(String expresión)** es el que va a evaluar la expresión y retornará el resultado de esta. Así mismo, utiliza el método **evaluar(String operador, String valor1, String valor2)**, especialmente, este método es el que va a registrar al cliente para poder realizar las operaciones aritméticas. Se puede observar, que, dependiendo del operador, invoca al método remoto correspondiente.



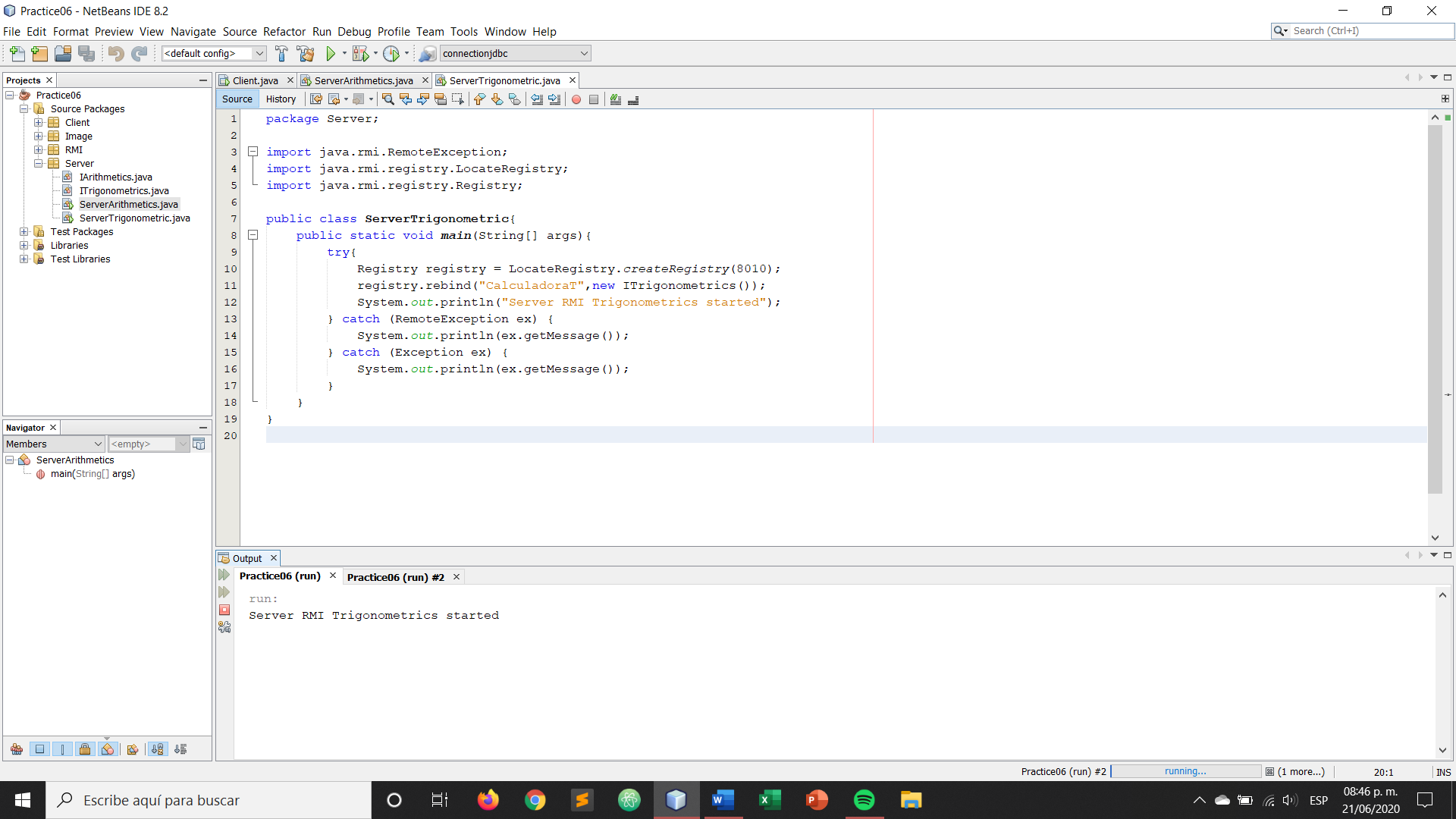


Básicamente, el único botón importante es el de calcular. Ya que, dentro de su Action Performed, registramos al cliente para poder evaluar las operaciones trigonométricas, posteriormente, evaluamos todas las operaciones de este tipo, esto con la finalidad de que la expresión tenga únicamente valores númericos.



FUNCIONAMIENTO DE LA APLICACIÓN

El primer paso, es ejecutar ambos servidores, para que registren los métodos a implementar y estén atentos a cualquier petición por parte del cliente.



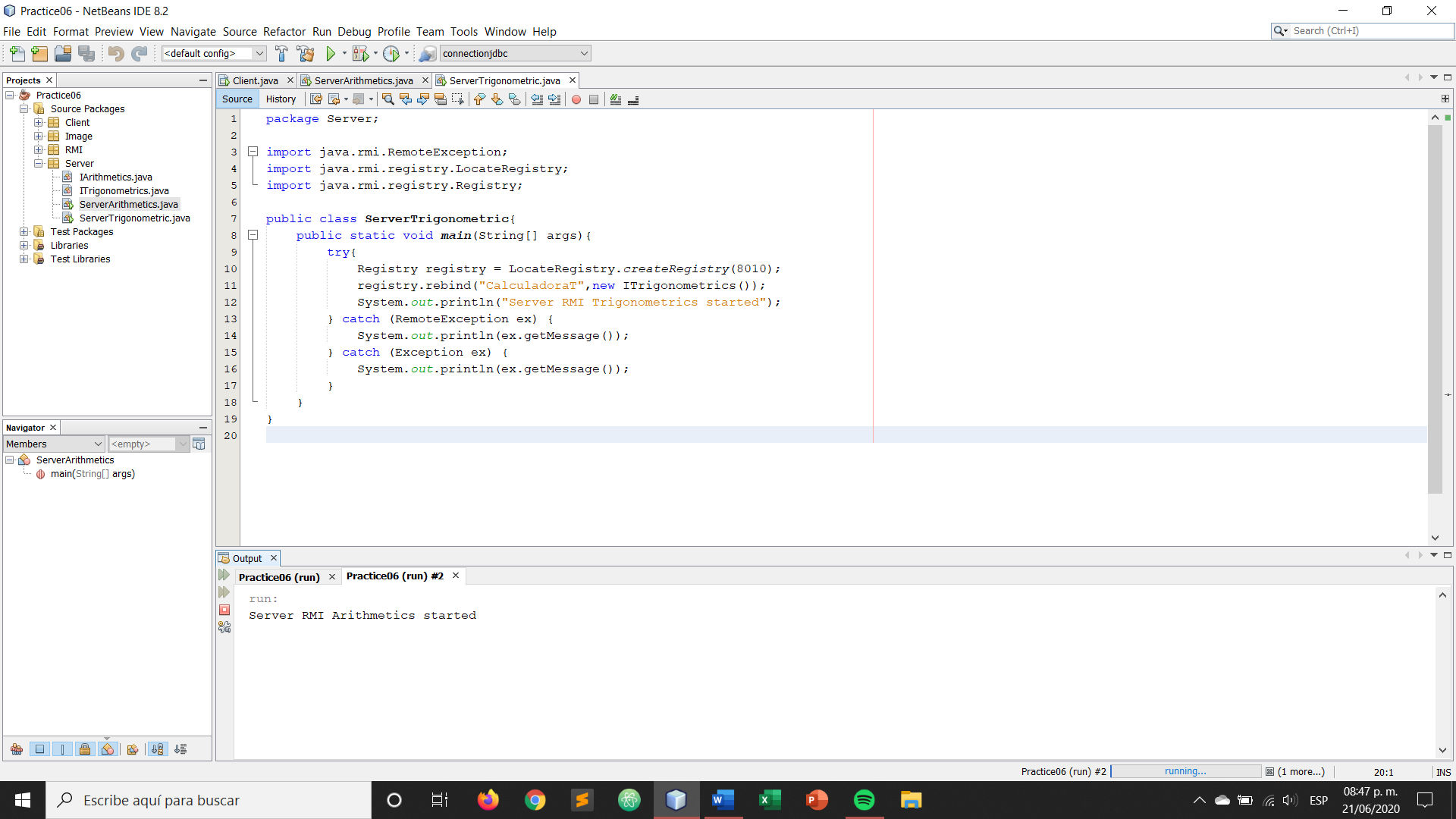


Imagen 1. Servidores ejecutándose.

Posteriormente, ejecutamos el cliente y se mostrará la ventana principal. Dentro de esta ventana, se puede observar que posee un JTextField para que el usuario introduzca la expresión que desea evaluar. Además, tiene 3 botones, el primero es para evaluar, el segundo es para limpiar el panel y el tercero para eliminar el contenido del JTextField.

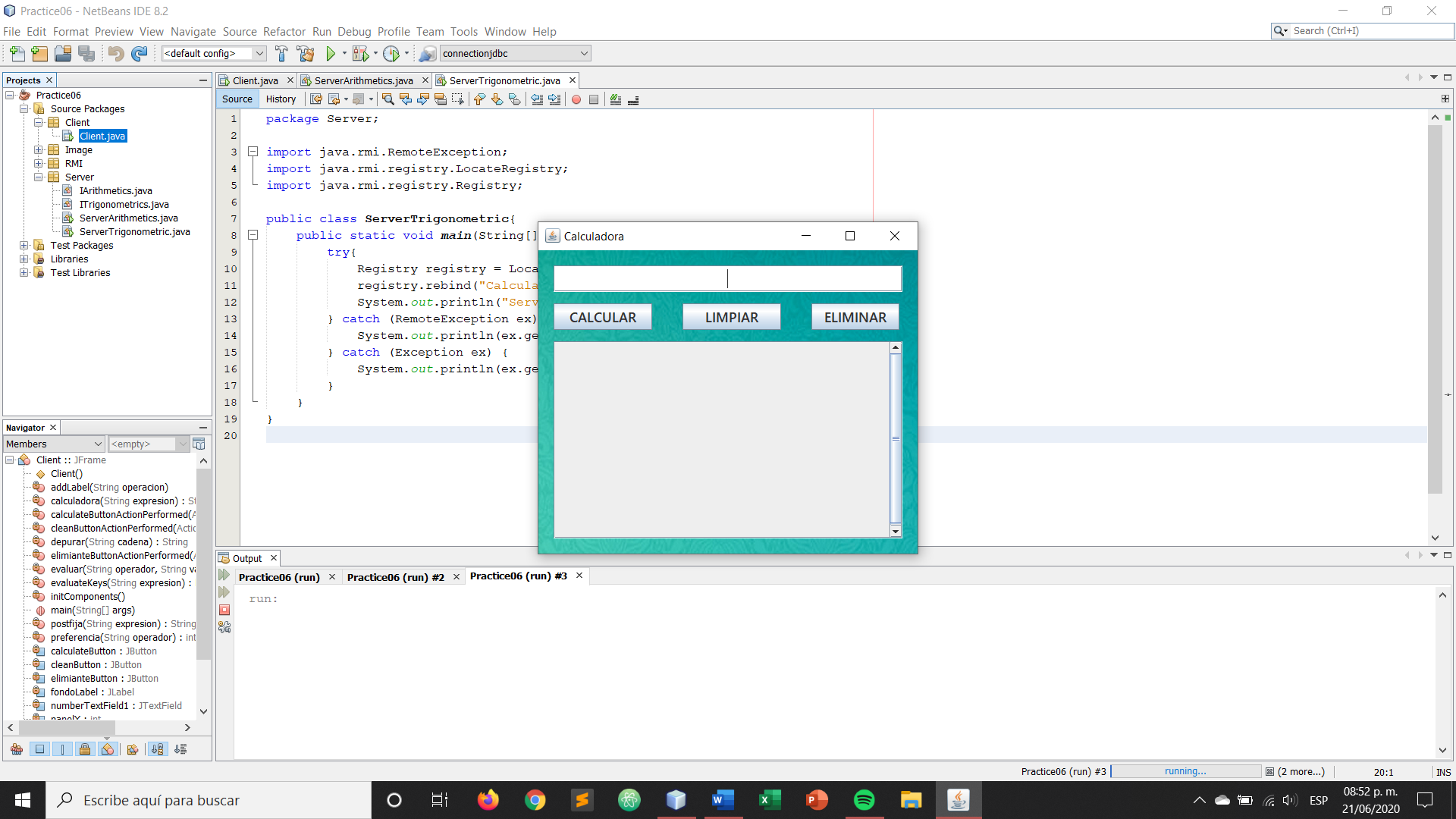


Imagen 2. Interfaz general.

Después, evaluamos las operaciones trigonométricas básicas y comparamos los resultados con los de la calculadora online de Google y observamos que son los mismo, por lo que las operaciones, se están realizando correctamente.

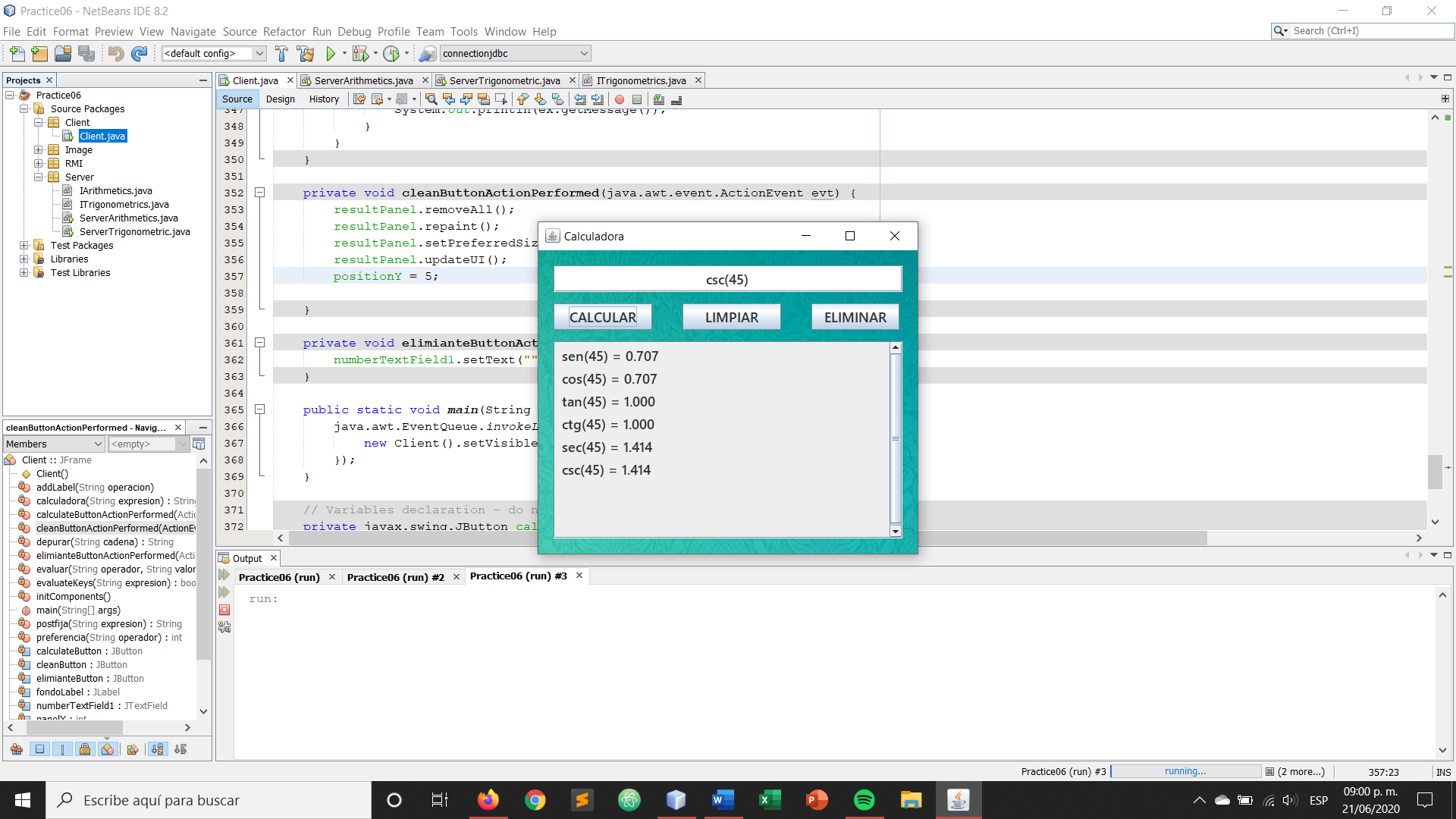
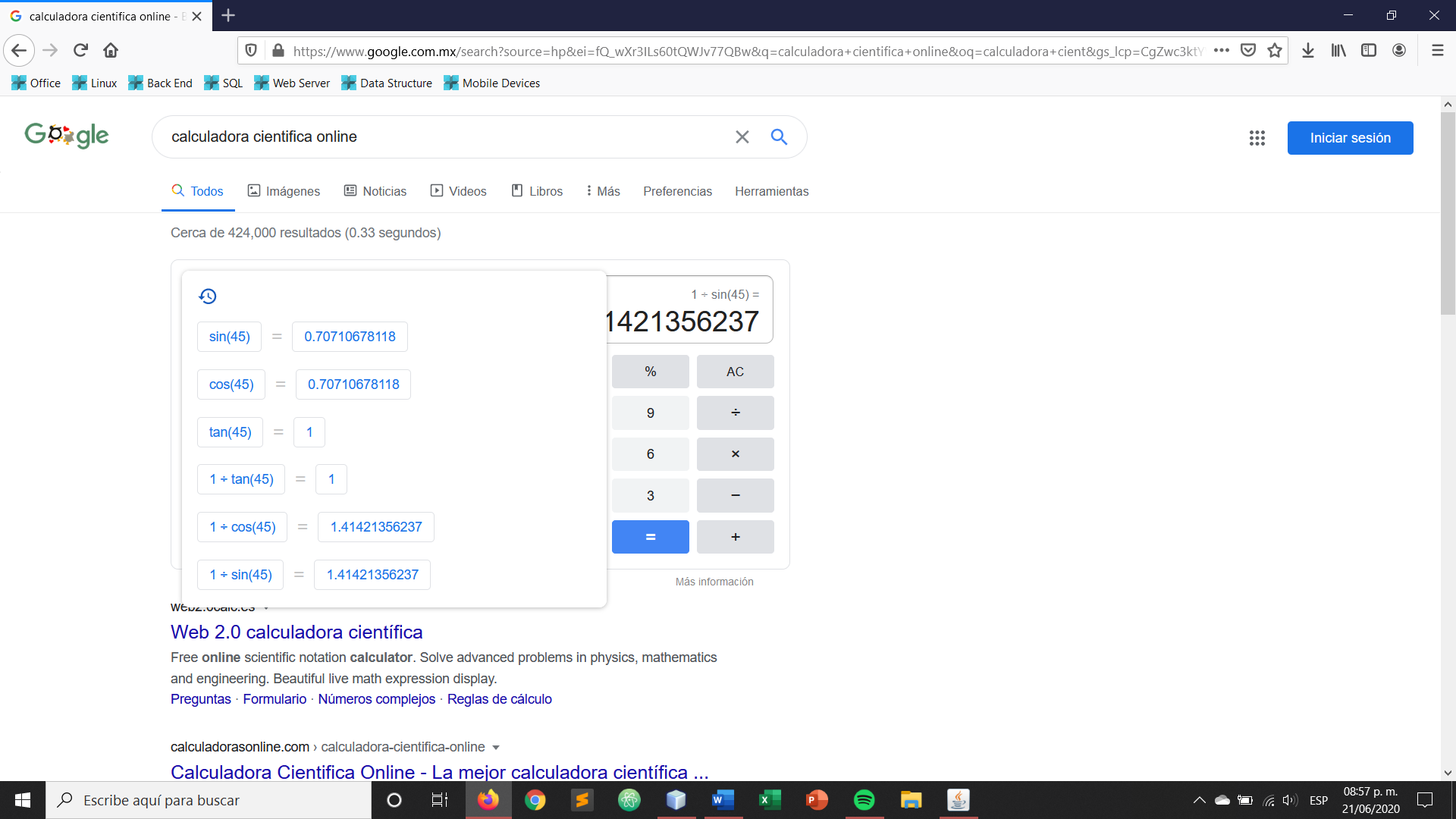


Imagen 3. Operaciones trigonométricas.

Además, a esta calculadora se le agrego la funcionalidad de evaluar los paréntesis. Es decir, evaluará que todos los paréntesis estén balanceados para poder calcular la expresión, en caso contrario, ni siquiera evaluará la expresión.

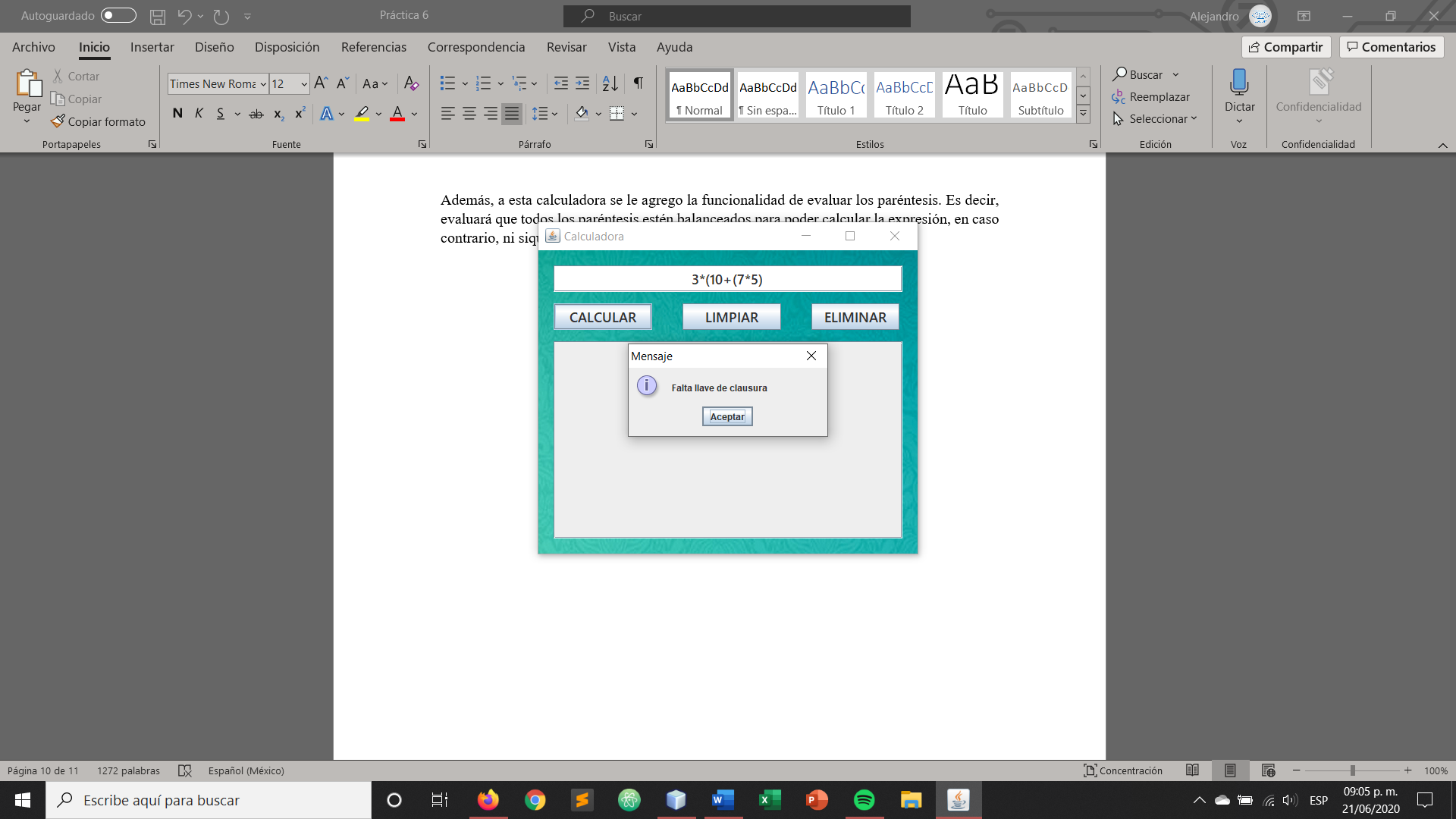
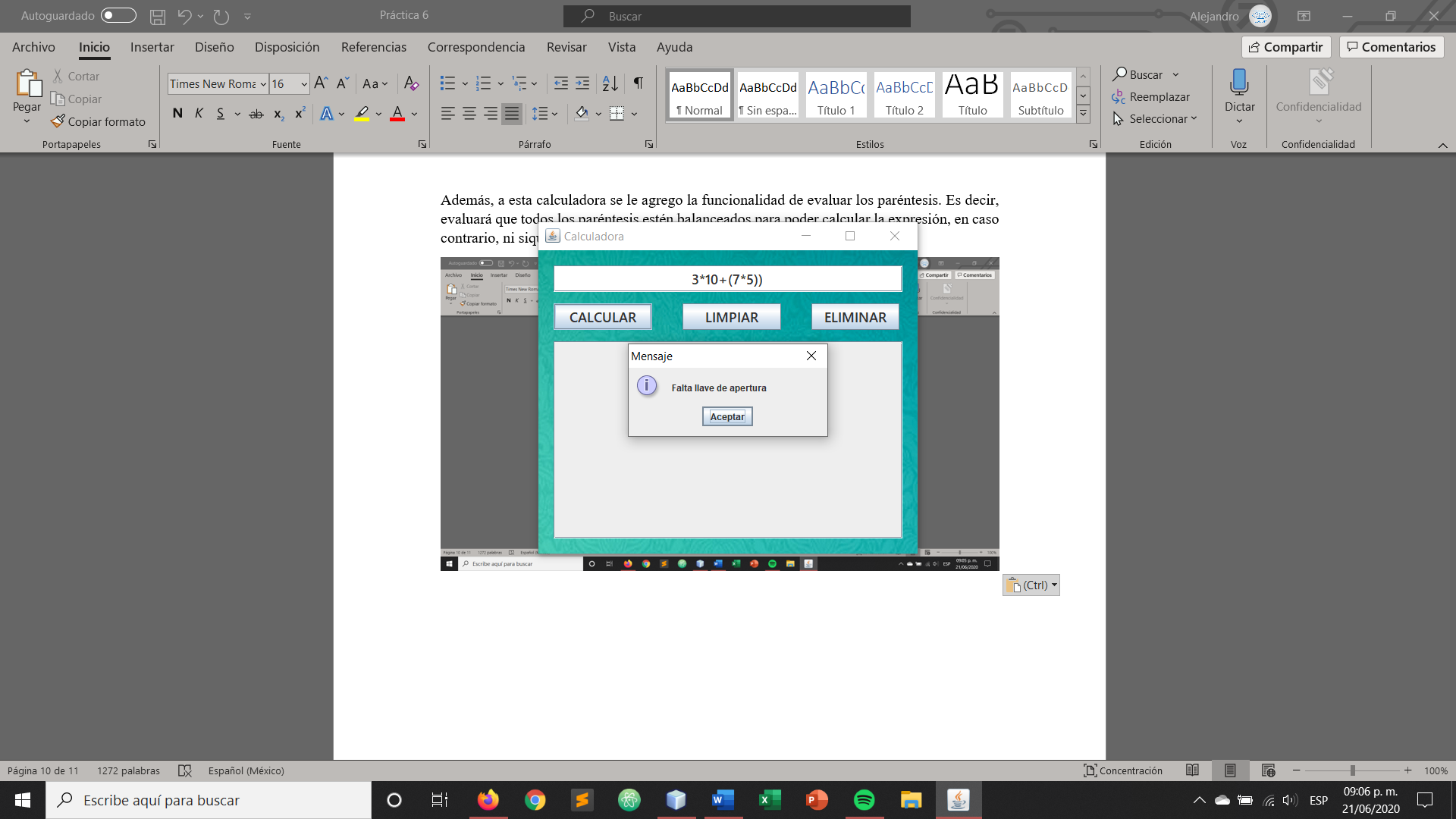
 

Imagen 4. Paréntesis balanceados.

Por último, realizaremos pruebas un poco más complejas para evaluar que el funcionamiento de la calculadora sea el correcto. Introduciremos 5 expresiones relativamente complejas y verificaremos el resultado.

ADVERTENCIA, EL ESTADO ACTUAL DE ESTA CALCULADORA ES RADIANES, EN CASO DE QUERER MODIFICAR A GRADOS, ACTUALIZAR LOS RMI.

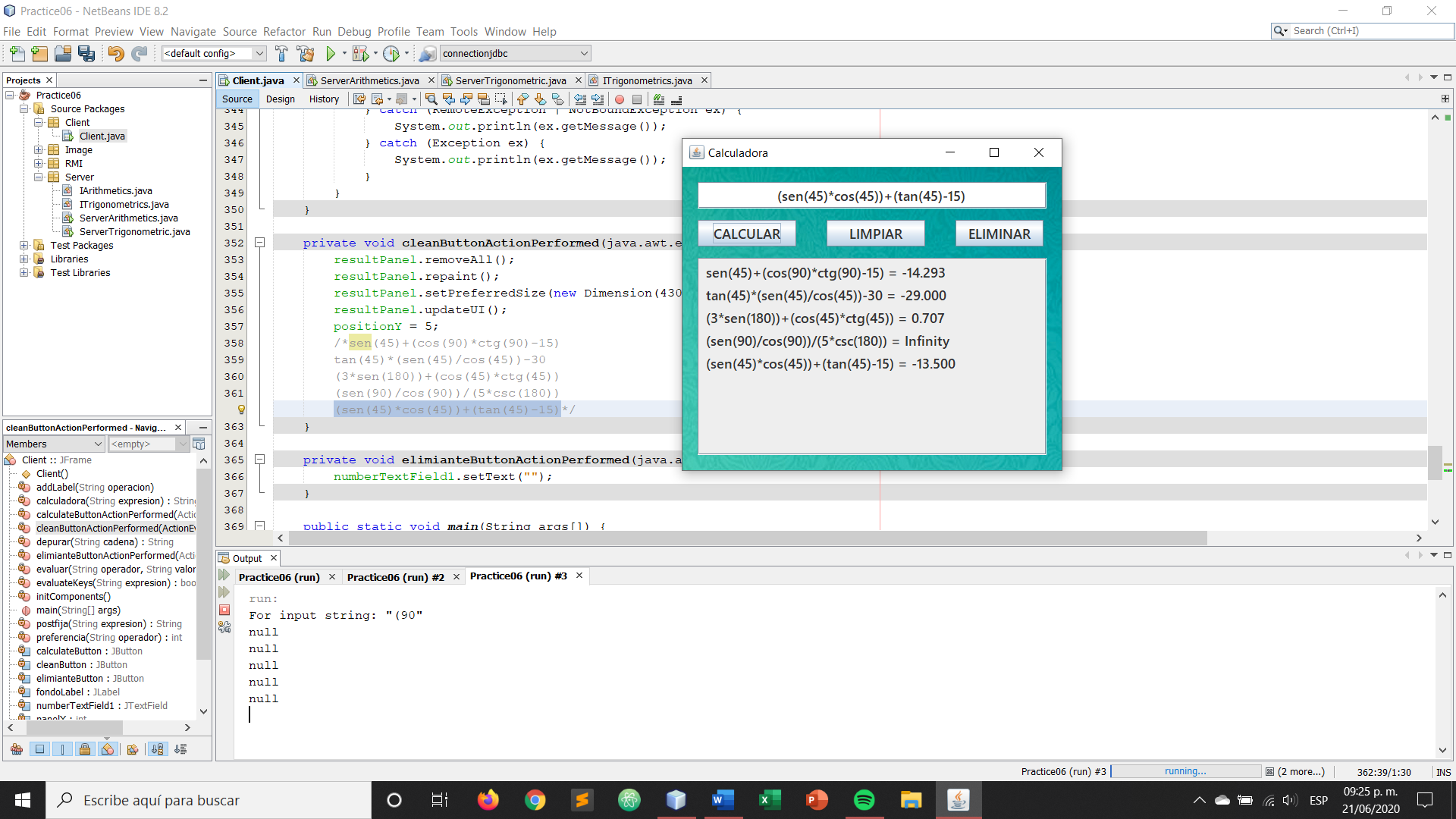


Imagen 5. Evaluación general.

CONCLUSIÓN

RMI es un sistema que nos permite el intercambio de objetos. Este intercambio se realiza de manera transparente de un espacio de dirección a otro, puesto que utiliza una técnica de serialización. Además, que nos permite el llamado de los métodos remotamente, sin tener la necesidad de tener los métodos localmente.

Debido a que los sistemas necesitan manejo de datos en sistemas distribuidos cuando estos residen en direcciones de distintos hosts, los métodos de invocación remota son una buena alternativa para solucionar estas necesidades.

RMI es un sistema de programación para la distribución e intercambio de datos entre distintas aplicaciones existentes en un entorno distribuido. Además, debido a las características de RMI que mencionamos al inicio del reporte, la seleccionamos como arquitectura o medio de compartir datos a través de la red.

Sin embargo, podemos mencionar otras características tales como la facilidad de usar y programar, ya que es un sistema totalmente de Java, además que permite realizar sistemas distribuidos orientados a objetos.