Imagen en blanco y negro

Descripción generada automáticamente con confianza baja

Laboratorio 8

Broker de Mensajes STOMP con WebSockets + HTML5 Canvas

url repositorio:

<https://github.com/ARSW2023-2/ARSW_Lab08.git>

Integrantes:

Luisa Fernanda Bermúdez Girón

Karol Daniela Ladino Ladino

Squad:

Inside Out

Profesor:

Javier Iván Toquica Barrera

Curso:

ARSW – 1

Fecha De Entrega:

3-11-2023

1. **Introducción**
2. **Desarrollo del laboratorio**

* Conectarse con un botón
* Publicar con eventos de mouse

|  |
| --- |
| var newpoint = JSON.parse(greeting.body);  addPointToCanvas(newpoint);  stompClient.send("/topic/newpoint", {}, JSON.stringify(pt)); |

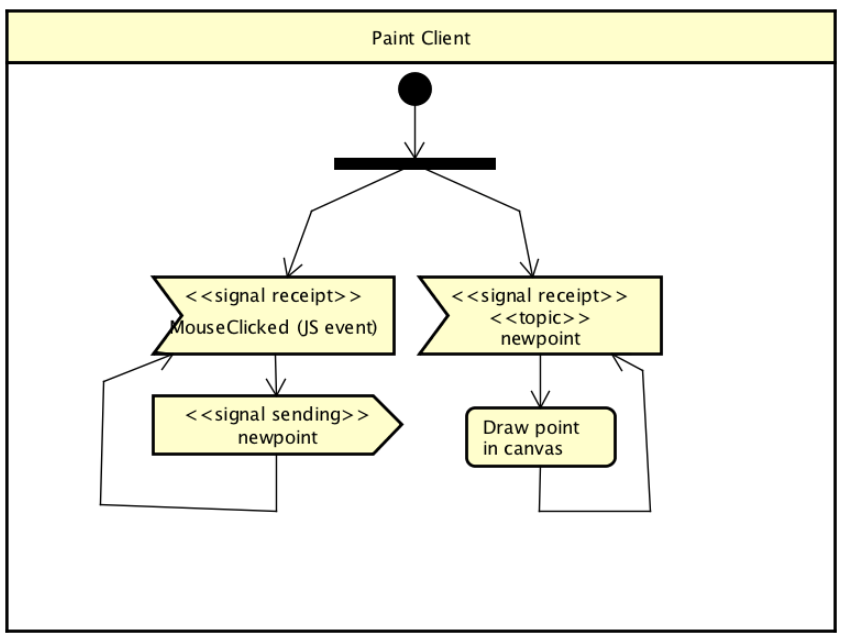
Este ejercicio se basa en la documentación oficial de SprinbBoot, para el manejo de WebSockets con STOMP.

En este repositorio se encuentra una aplicación SpringBoot que está configurado como Broker de mensajes, de forma similar a lo mostrado en la siguiente figura:

En este caso, el manejador de mensajes asociado a "/app" aún no está configurado, pero sí lo está el broker '/topic'. Como mensaje, se usarán puntos, pues se espera que esta aplicación permita progragar eventos de dibujo de puntos generados por los diferentes clientes.

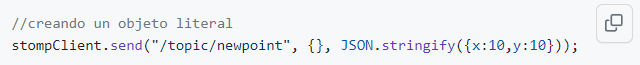
**Parte I**

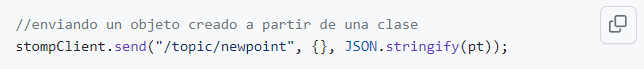
Para las partes I y II, usted va a implementar una herramienta de dibujo colaborativo Web, basada en el siguiente diagrama de actividades:



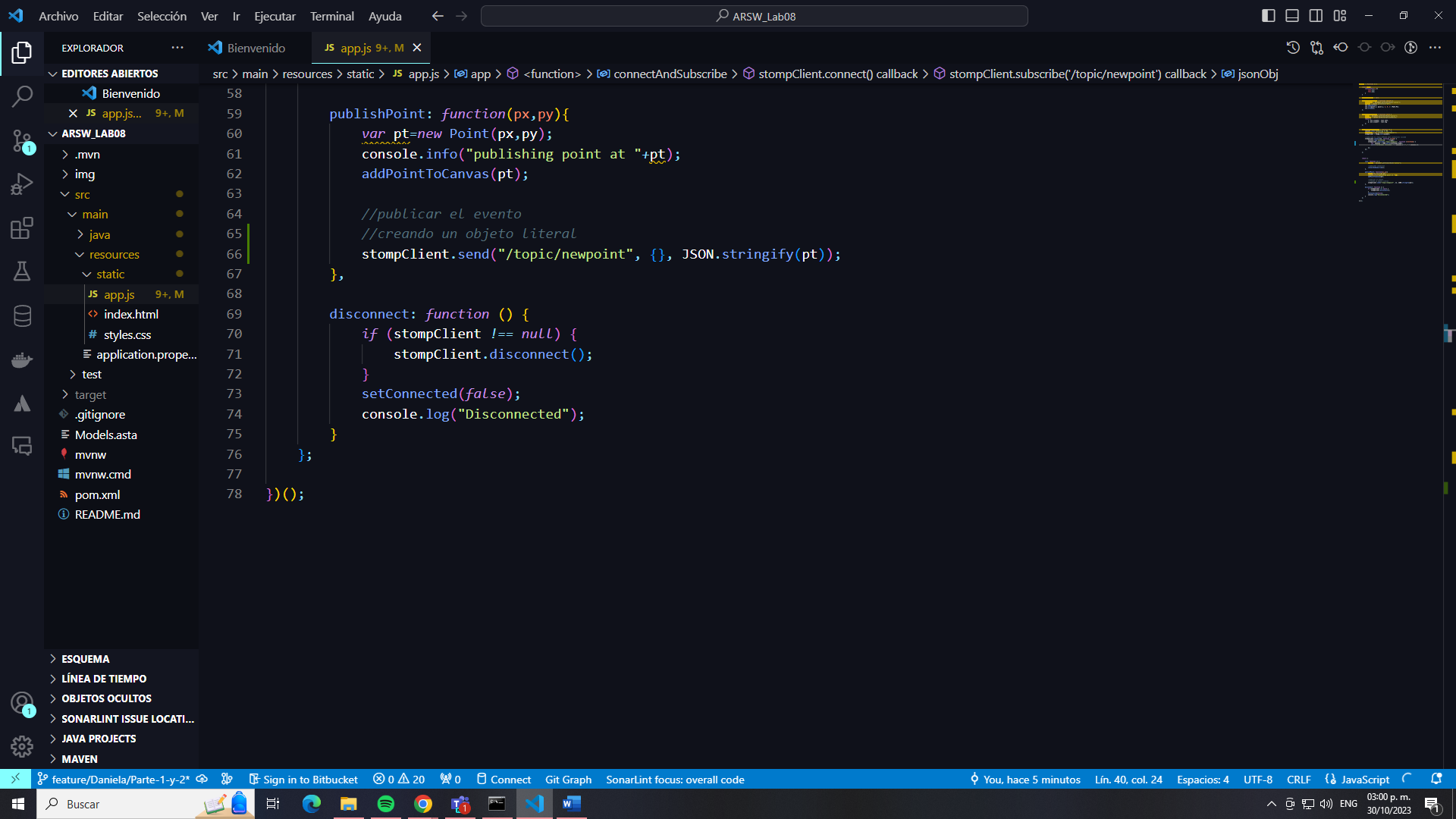
Para esto, realice lo siguiente:

1. Haga que la aplicación HTML5/JS al ingresarle en los campos de X y Y, además de graficarlos, los publique en el tópico: /topic/newpoint . Para esto tenga en cuenta (1) usar el cliente STOMP creado en el módulo de JavaScript y (2) enviar la representación textual del objeto JSON (usar JSON.stringify). Por ejemplo:





* Modificamos en app.js la función publishPoint



1. Dentro del módulo JavaScript modifique la función de conexión/suscripción al WebSocket, para que la aplicación se suscriba al tópico "/topic/newpoint" (en lugar del tópico /TOPICOXX). Asocie como 'callback' de este suscriptor una función que muestre en un mensaje de alerta (alert()) el evento recibido. Como se sabe que en el tópico indicado se publicarán sólo puntos, extraiga el contenido enviado con el evento (objeto JavaScript en versión de texto), conviértalo en objeto JSON, y extraiga de éste sus propiedades (coordenadas X y Y). Para extraer el contenido del evento use la propiedad 'body' del mismo, y para convertirlo en objeto, use JSON.parse. Por ejemplo:



* Modificamos en app.js la función connectAndSubscribe

Captura de pantalla de computadora

Descripción generada automáticamente

1. Compile y ejecute su aplicación. Abra la aplicación en varias pestañas diferentes (para evitar problemas con el caché del navegador, use el modo 'incógnito' en cada prueba).

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

1. Ingrese los datos, ejecute la acción del botón, y verifique que en todas las pestañas se haya lanzado la alerta con los datos ingresados.

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

1. Haga commit de lo realizado, para demarcar el avance de la parte 2.



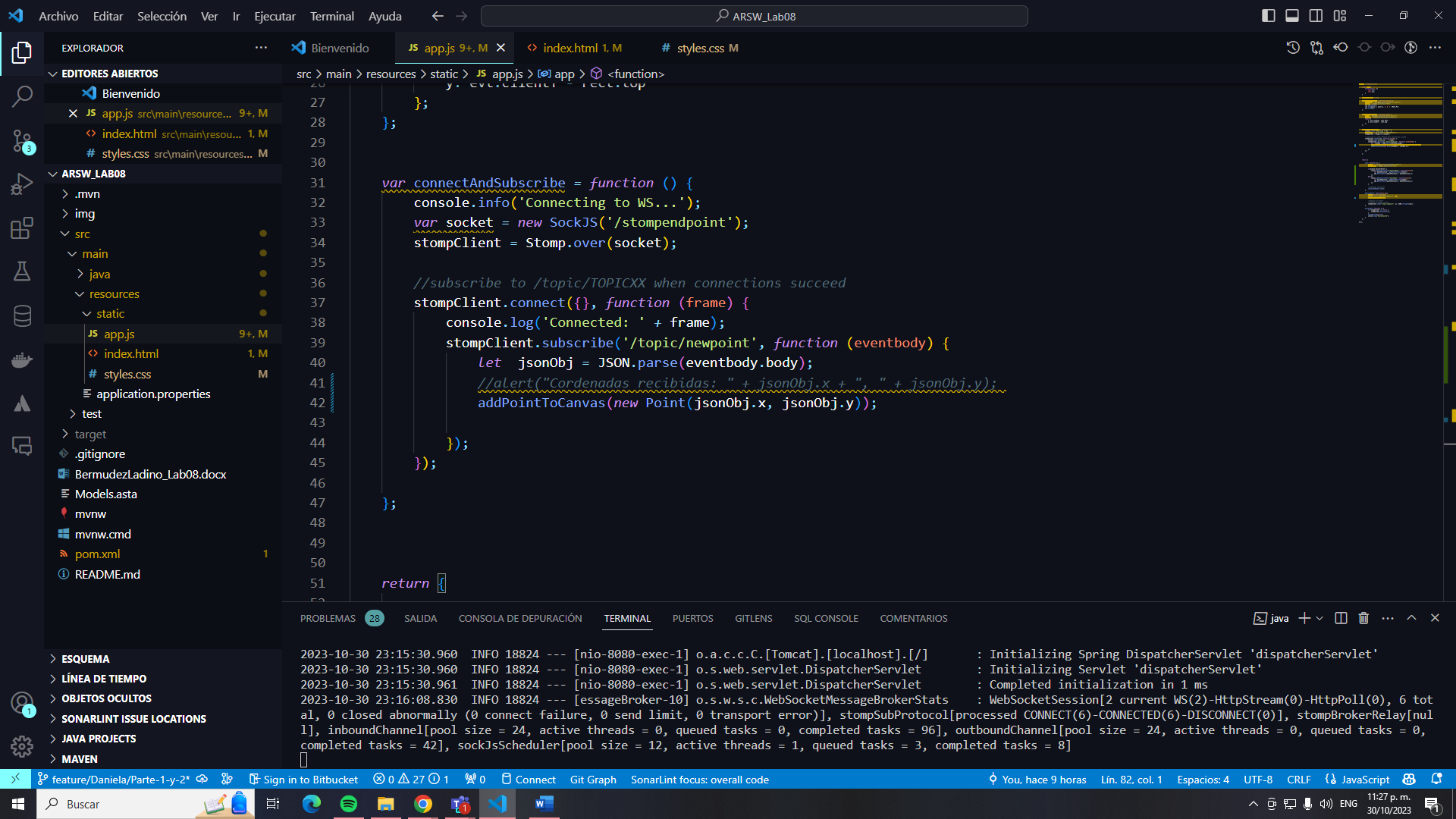
* Visualizar los commits del repositorio.

**Parte II**

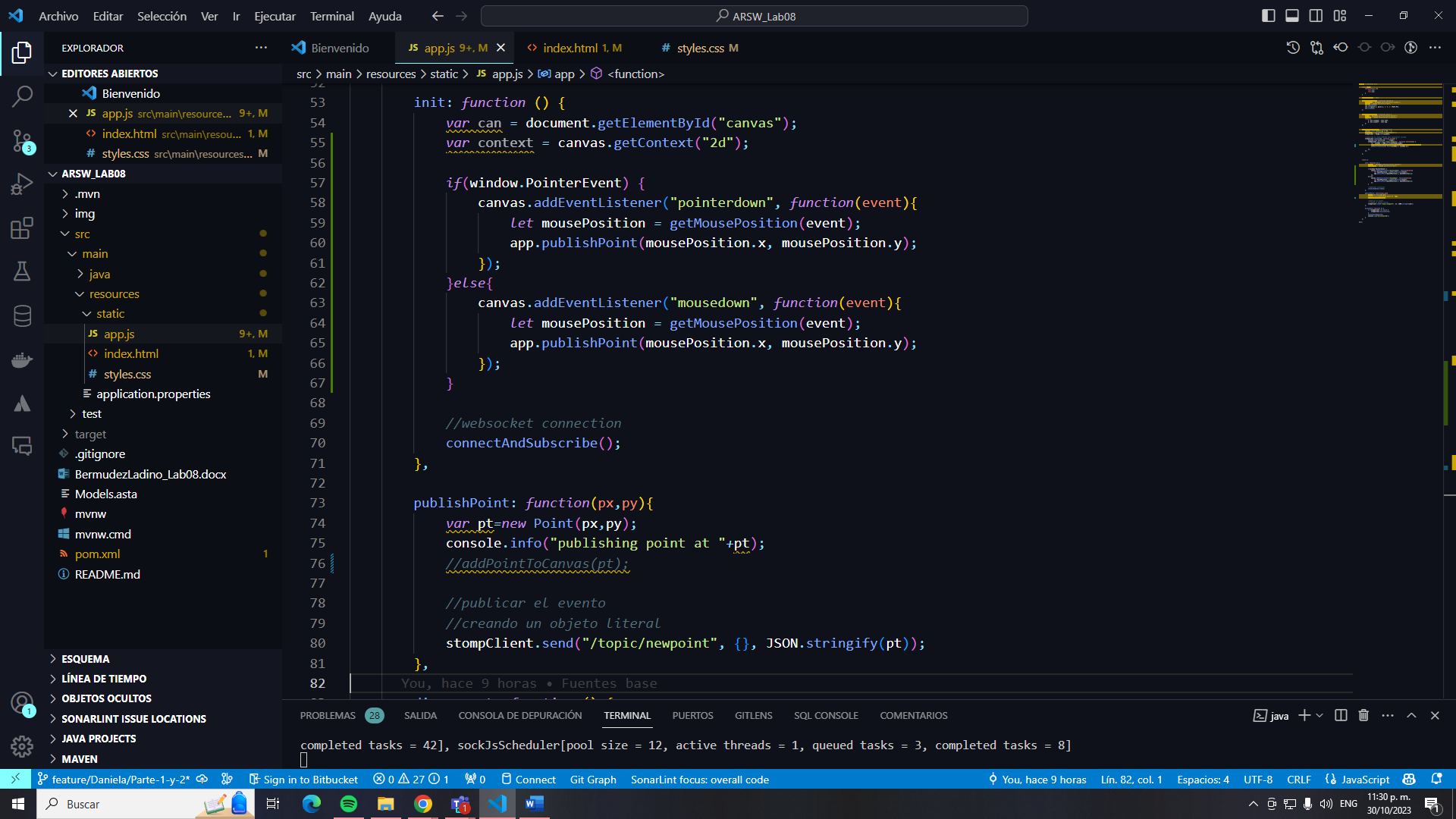
Para hacer más útil la aplicación, en lugar de capturar las coordenadas con campos de formulario, las va a capturar a través de eventos sobre un elemento de tipo <canvas>. De la misma manera, en lugar de simplemente mostrar las coordenadas enviadas en los eventos a través de 'alertas', va a dibujar dichos puntos en el mismo canvas. Haga uso del mecanismo de captura de eventos de mouse/táctil usado en ejercicios anteriores con este fin.

1. Haga que el 'callback' asociado al tópico /topic/newpoint en lugar de mostrar una alerta, dibuje un punto en el canvas en las coordenadas enviadas con los eventos recibidos. Para esto puede dibujar un círculo de radio 1.

* Modificamos en app.js la función connectAndSubscribe.



* Modificamos en app.js la función init y publishPoint.



1. Ejecute su aplicación en varios navegadores (y si puede en varios computadores, accediendo a la aplicación mendiante la IP donde corre el servidor). Compruebe que a medida que se dibuja un punto, el mismo es replicado en todas las instancias abiertas de la aplicación.

Imagen que contiene Calendario

Descripción generada automáticamente

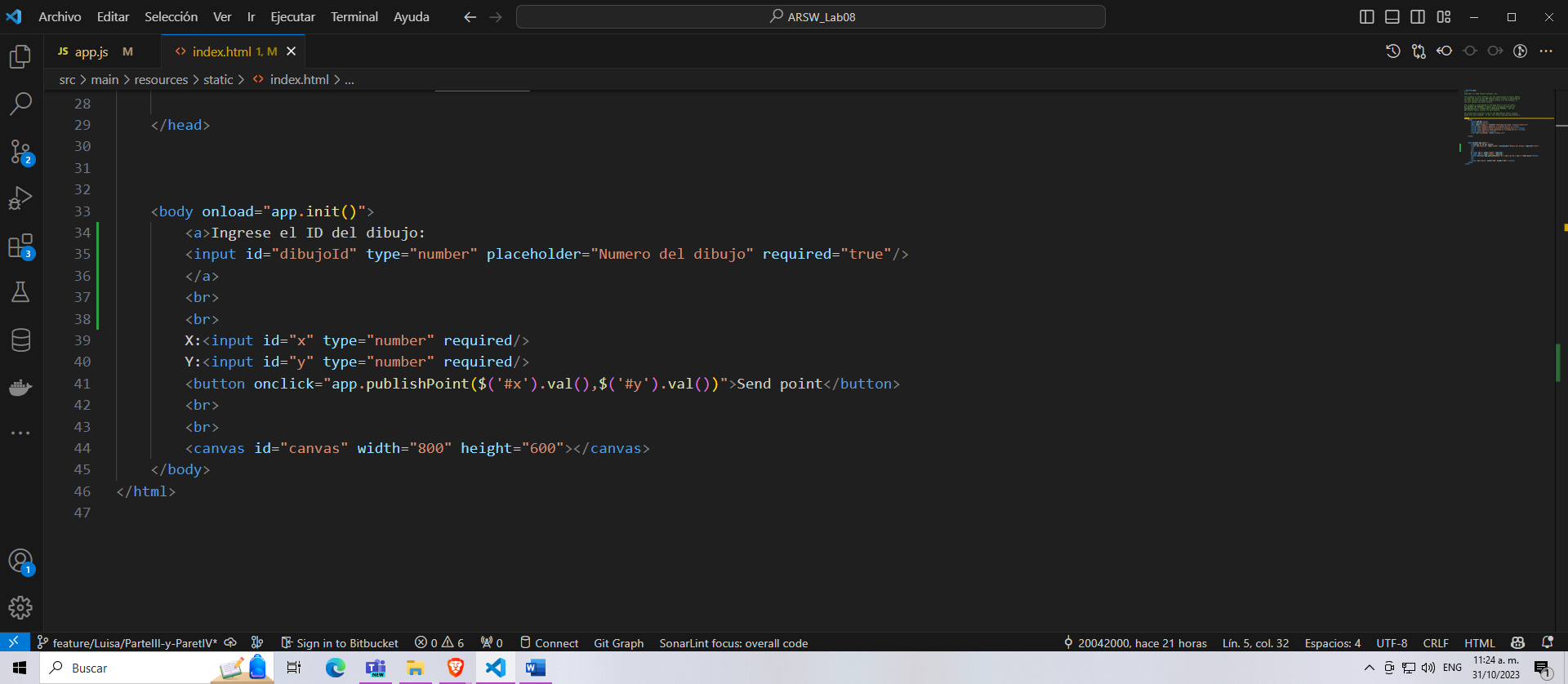
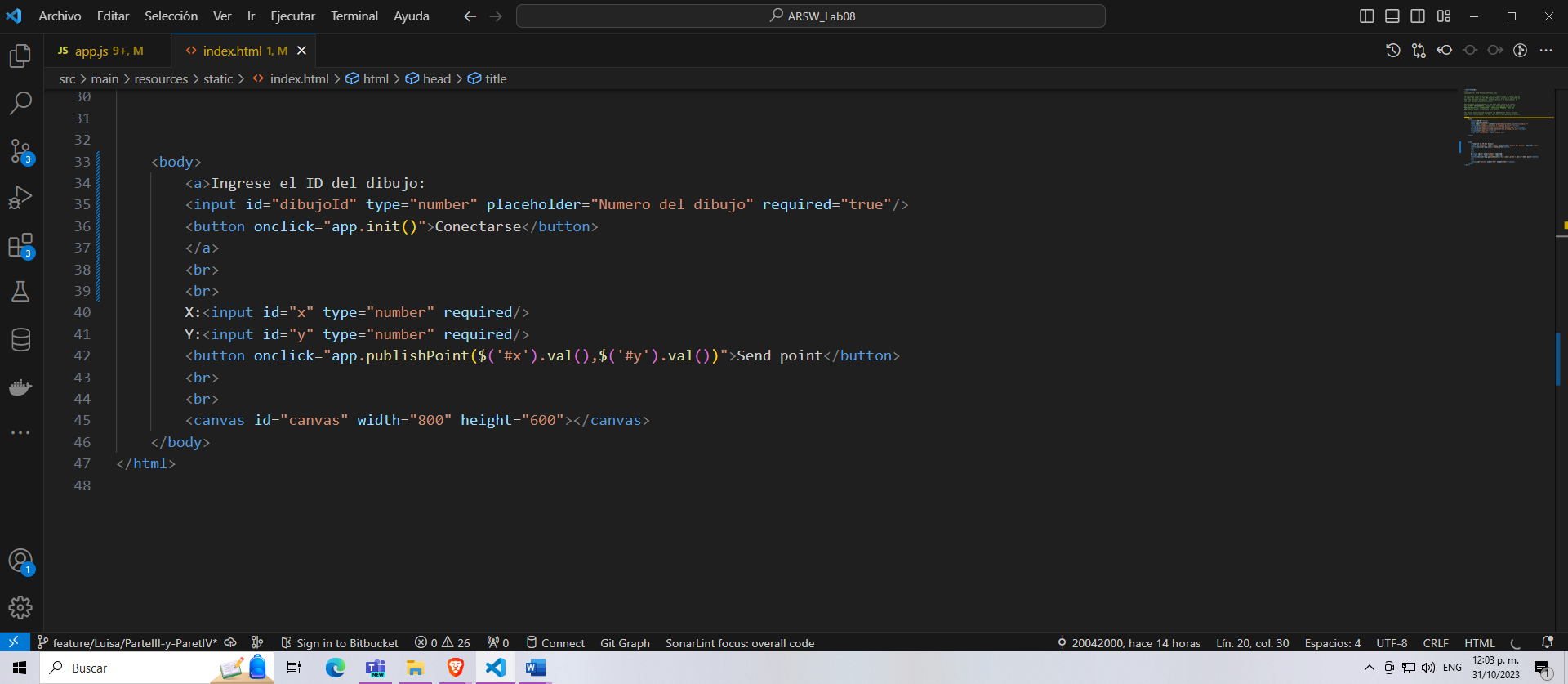
1. Haga commit de lo realizado, para marcar el avance de la parte 2.

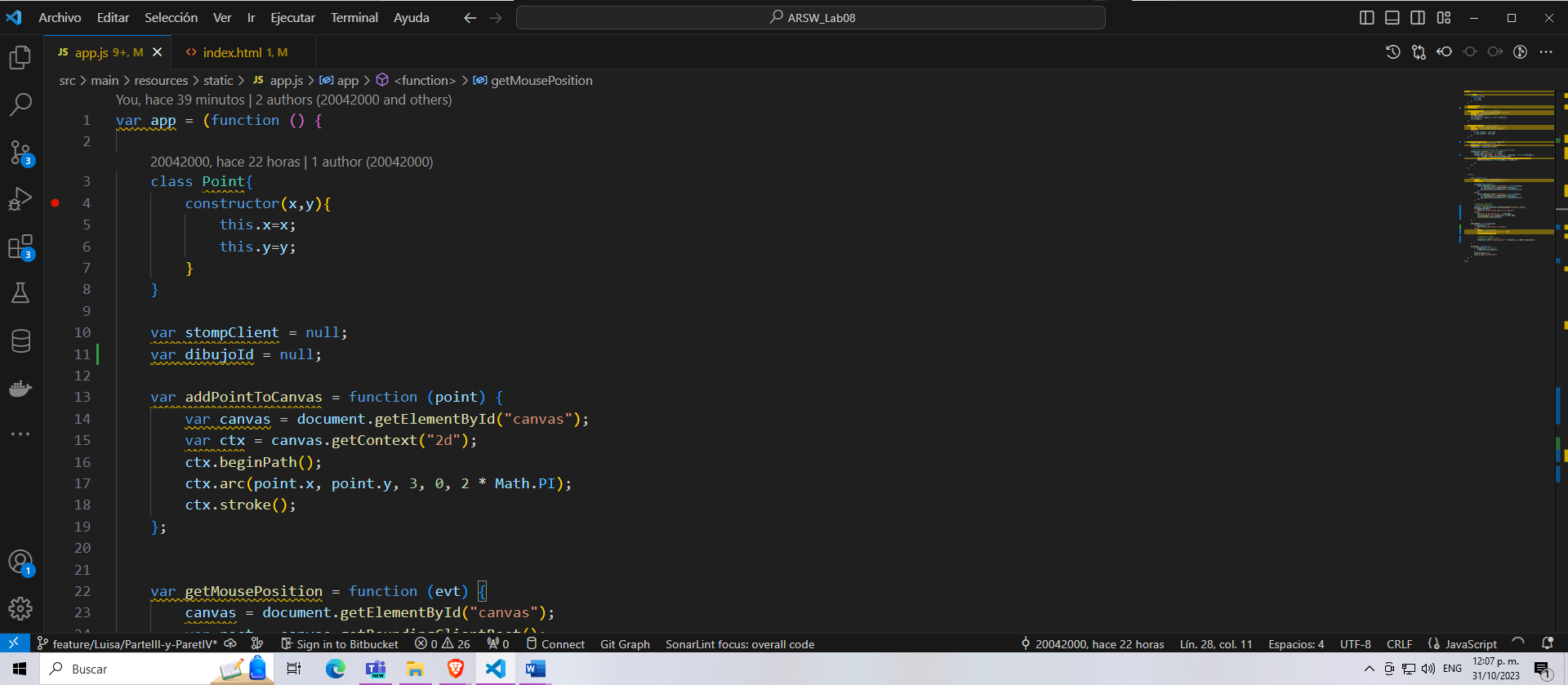


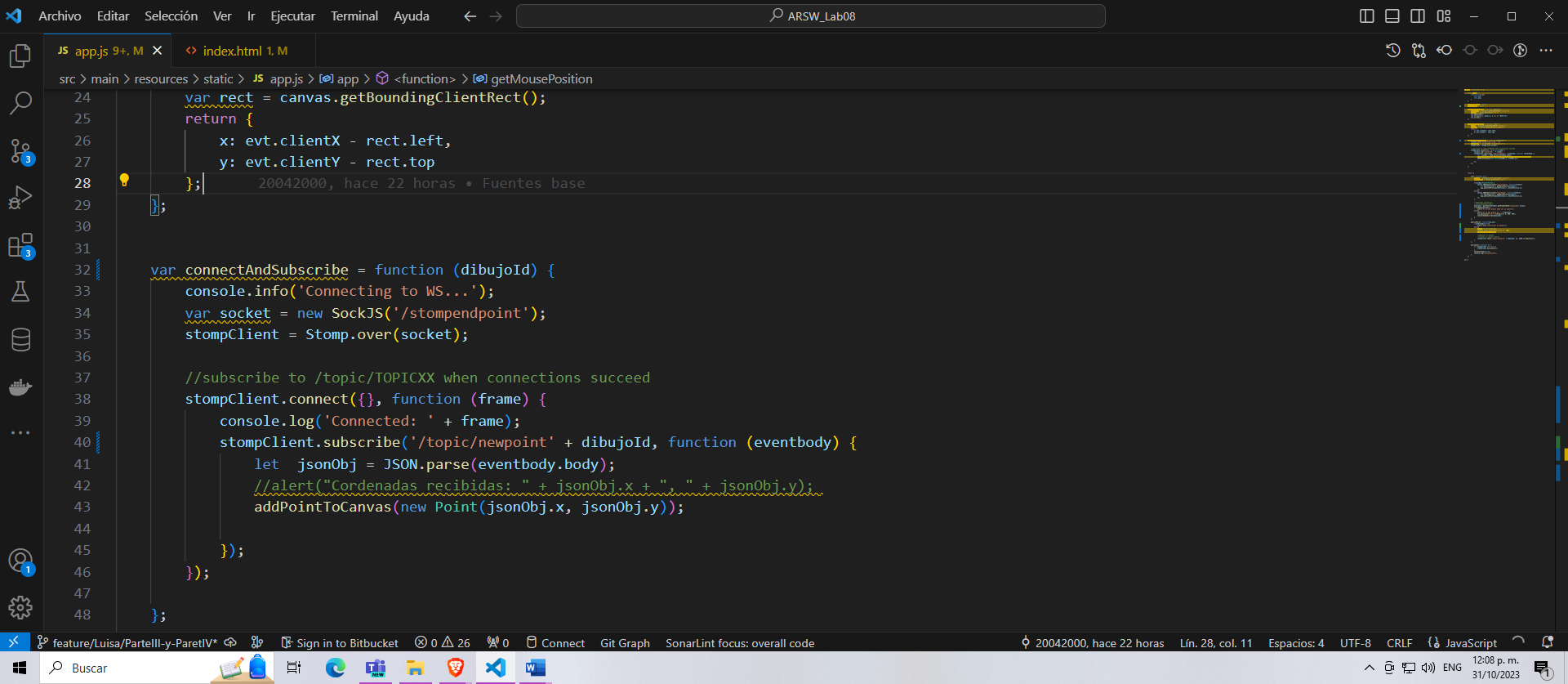
* Visualizar los commits del repositorio.

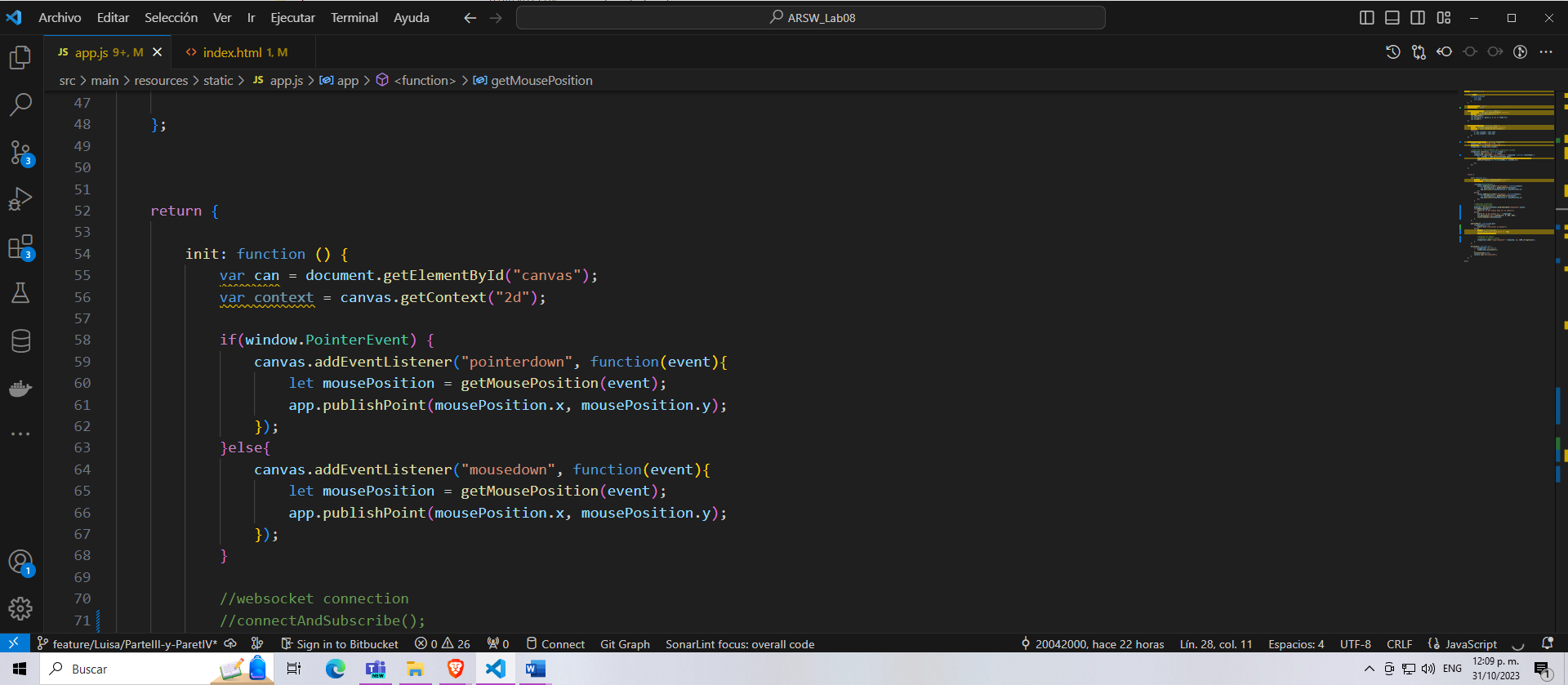
**Parte III**

Ajuste la aplicación anterior para que pueda manejar más de un dibujo a la vez, manteniendo tópicos independientes. Para esto:

* + - 1. Agregue un campo en la vista, en el cual el usuario pueda ingresar un número. El número corresponderá al identificador del dibujo que se creará.
* Modificamos index.html.
  + - 1. Modifique la aplicación para que, en lugar de conectarse y suscribirse automáticamente (en la función init()), lo haga a través de botón 'conectarse'. Éste, al oprimirse debe realizar la conexión y suscribir al cliente a un tópico que tenga un nombre dinámico, asociado el identificador ingresado, por ejemplo: /topic/newpoint.25, topic/newpoint.80, para los dibujos 25 y 80 respectivamente.
* Modificamos index.html.
* Modificamos app.js







Captura de pantalla de computadora

Descripción generada automáticamente

* + - 1. De la misma manera, haga que las publicaciones se realicen al tópico asociado al identificador ingresado por el usuario.

Captura de pantalla de computadora

Descripción generada automáticamente

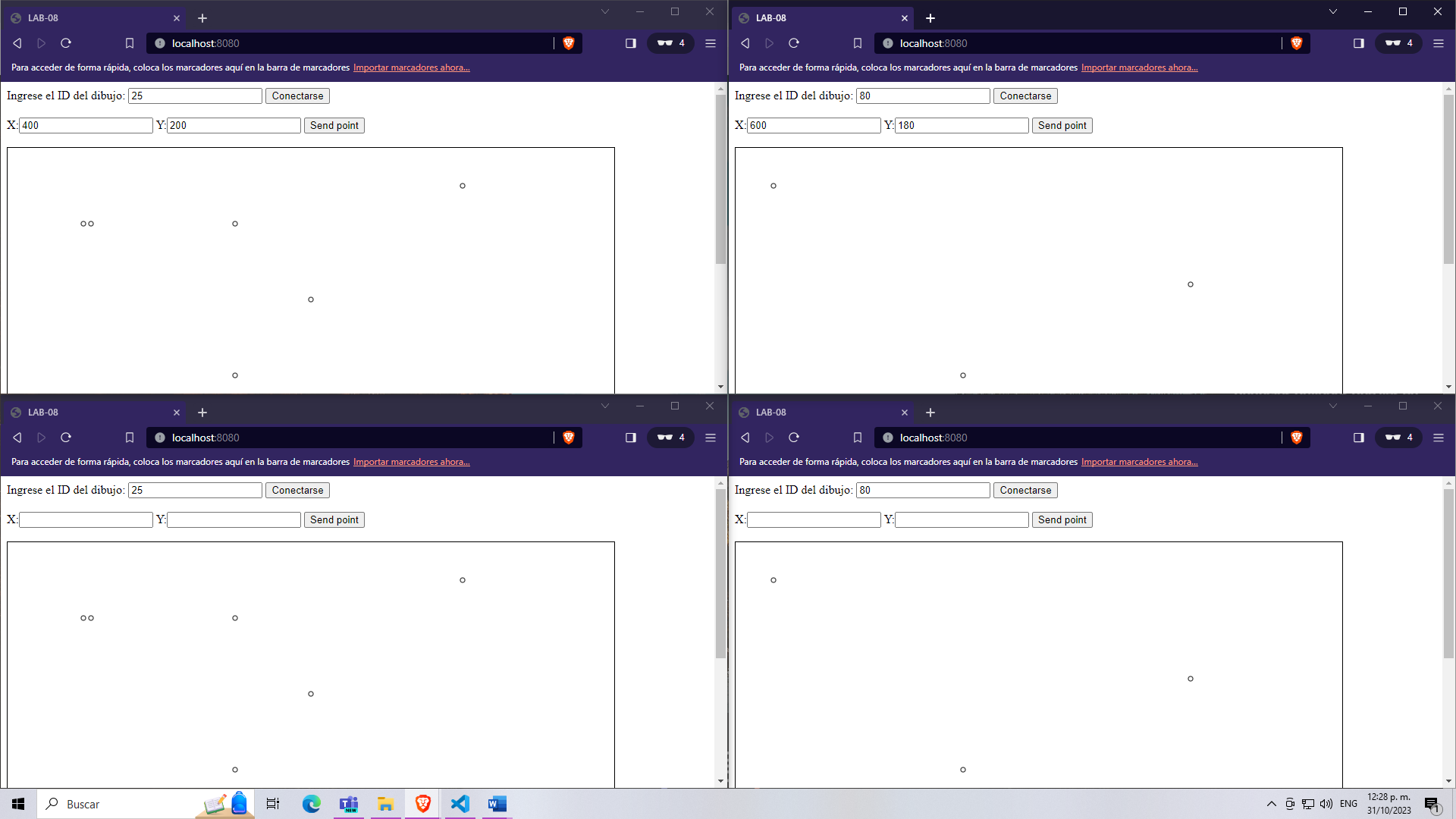
* + - 1. Rectifique que se puedan realizar dos dibujos de forma independiente, cada uno de éstos entre dos o más clientes.



* Visualizar los commits del repositorio.

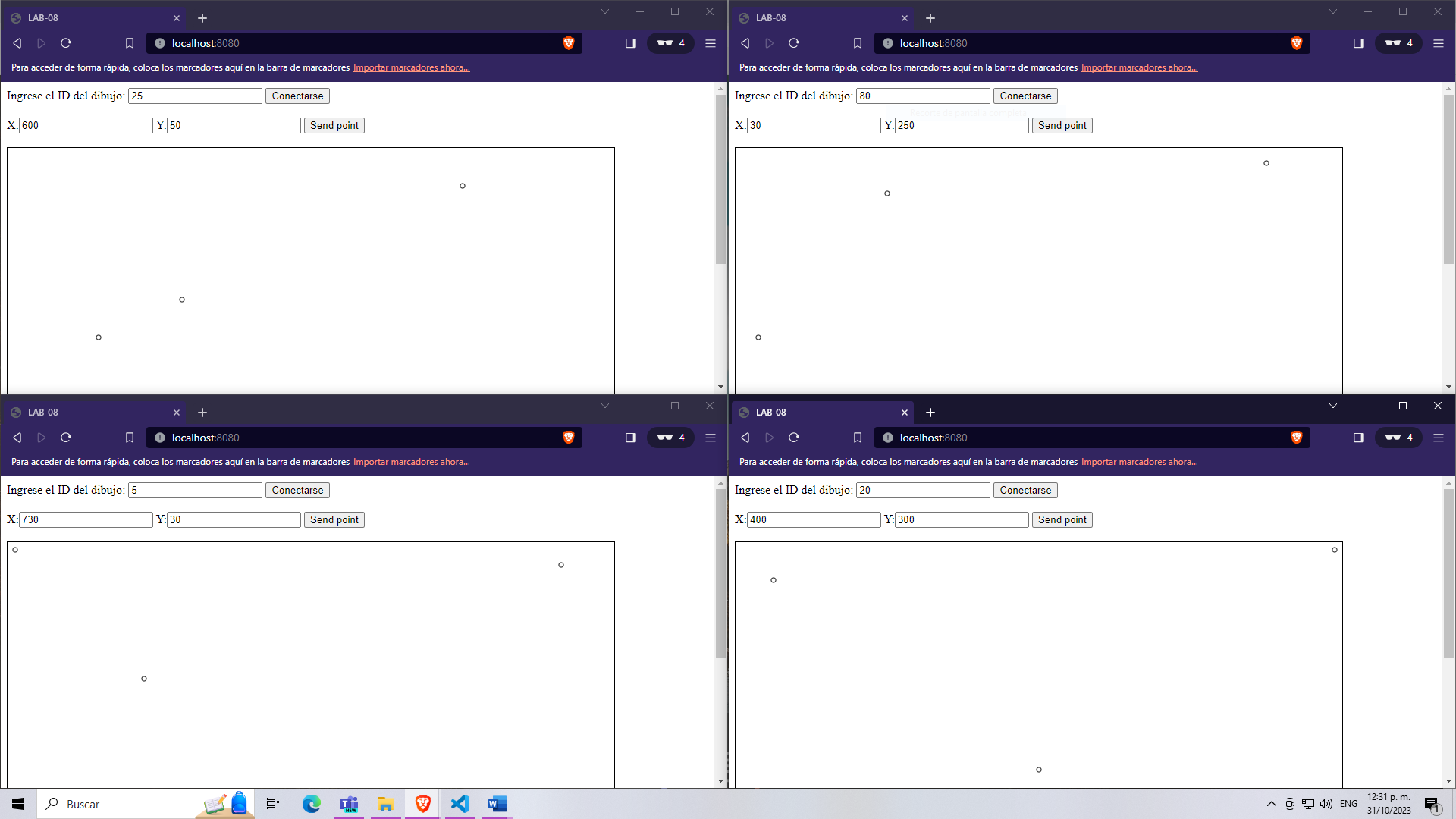
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



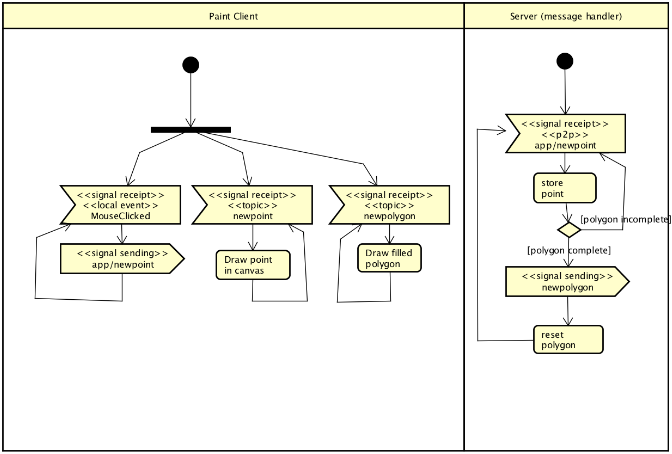
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



**Parte IV**

Para la parte IV, usted va a implementar una versión extendida del modelo de actividades y eventos anterior, en la que el servidor (que hasta ahora sólo fungía como Broker o MOM -Message Oriented Middleware-) se volverá también suscriptor de ciertos eventos, para a partir de los mismos agregar la funcionalidad de 'dibujo colaborativo de polígonos':



Para esto, se va a hacer una configuración alterna en la que, en lugar de que se propaguen los mensajes 'newpoint.{numdibujo}' entre todos los clientes, éstos sean recibidos y procesados primero por el servidor, de manera que se pueda decidir qué hacer con los mismos.

Para ver cómo manejar esto desde el manejador de eventos STOMP del servidor, revise puede revisar la documentación de Spring.

1. Cree una nueva clase que haga el papel de 'Controlador' para ciertos mensajes STOMP (en este caso, aquellos enviados a través de "/app/newpoint.{numdibujo}"). A este controlador se le inyectará un bean de tipo SimpMessagingTemplate, un Bean de Spring que permitirá publicar eventos en un determinado tópico. Por ahora, se definirá que cuando se intercepten los eventos enviados a "/app/newpoint.{numdibujo}" (que se supone deben incluir un punto), se mostrará por pantalla el punto recibido, y luego se procederá a reenviar el evento al tópico al cual están suscritos los clientes "/topic/newpoint".

Texto

Descripción generada automáticamente

1. Ajuste su cliente para que, en lugar de publicar los puntos en el tópico /topic/newpoint.{numdibujo}, lo haga en /app/newpoint.{numdibujo}. Ejecute de nuevo la aplicación y rectifique que funcione igual, pero ahora mostrando en el servidor los detalles de los puntos recibidos.
2. Una vez rectificado el funcionamiento, se quiere aprovechar este 'interceptor' de eventos para cambiar ligeramente la funcionalidad:
   1. Se va a manejar un nuevo tópico llamado '/topic/newpolygon.{numdibujo}', en donde el lugar de puntos, se recibirán objetos javascript que tengan como propiedad un conjunto de puntos.
   2. El manejador de eventos de /app/newpoint.{numdibujo}, además de propagar los puntos a través del tópico '/topic/newpoints', llevará el control de los puntos recibidos(que podrán haber sido dibujados por diferentes clientes). Cuando se completen tres o más puntos, publicará el polígono en el tópico '/topic/newpolygon'. Recuerde que esto se realizará concurrentemente, de manera que REVISE LAS POSIBLES CONDICIONES DE CARRERA!. También tenga en cuenta que desde el manejador de eventos del servidor se tendrán N dibujos independientes!.
   3. El cliente, ahora también se suscribirá al tópico '/topic/newpolygon'. El 'callback' asociado a la recepción de eventos en el mismo debe, con los datos recibidos, dibujar un polígono, tal como se muestran en ese ejemplo.
   4. Verifique la funcionalidad: igual a la anterior, pero ahora dibujando polígonos cada vez que se agreguen cuatro puntos.
3. A partir de los diagramas dados en el archivo ASTAH incluido, haga un nuevo diagrama de actividades correspondiente a lo realizado hasta este punto, teniendo en cuenta el detalle de que ahora se tendrán tópicos dinámicos para manejar diferentes dibujos simultáneamente.
4. Haga commit de lo realizado.



1. **Conclusiones**