



## Laboratorio 8

Broker de Mensajes STOMP con WebSockets + HTML5 Canvas

url repositorio:

[https://github.com/ARSW2023-2/ARSW\\_Lab08.git](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

## I. Introducción

En este laboratorio, se creará una aplicación web colaborativa que utiliza tecnologías como STOMP, WebSockets, HTML5 Canvas y JavaScript. El objetivo principal de esta aplicación es habilitar a múltiples usuarios para colaborar en la creación de dibujos en un lienzo compartido. Además, la aplicación aprovechará SpringBoot como un broker de mensajes para facilitar la comunicación en tiempo real entre los participantes.

## II. 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));
```

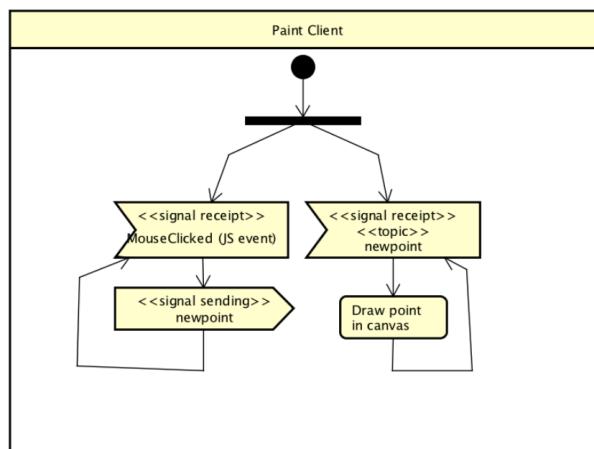
Este ejercicio se basa en la documentación oficial de SpringBoot, para el manejo de WebSockets con STOMP.

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:

```
//creando un objeto literal  
stompClient.send("/topic/newpoint", {}, JSON.stringify({x:10,y:10}));
```

```
//enviando un objeto creado a partir de una clase  
stompClient.send("/topic/newpoint", {}, JSON.stringify(pt));
```

- Modificamos en app.js la función publishPoint

The screenshot shows an IDE interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "EDIFICADORES ABIERTOS". It includes a "Bienvenido" folder, an "ARSW\_LAB08" folder containing ".mvn", "src", "main", "java", "resources", and "static" folders, and files like "index.html", "styles.css", "application.properties", "test", ".gitignore", "Models.java", "mvnw", "mvnw.cmd", "pom.xml", and "README.md".
- Code Editor:** The main editor window displays the "app.js" file. The code is as follows:

```
publishPoint: function(px,py){  
    var pt=new Point(px,py);  
    console.info("publishing point at "+pt);  
    addPointToCanvas(pt);  
  
    //publicar el evento  
    //creando un objeto literal  
    stompClient.send("/topic/newpoint", {}, JSON.stringify(pt));  
},  
  
disconnect: function () {  
    if (stompClient != null) {  
        stompClient.disconnect();  
    }  
    setConnected(false);  
    console.log("Disconnected");  
}
```

- Bottom Status Bar:** Shows information such as "You, hace 5 minutos", "Lis. 40, col. 24", "Espacio: 4", "UTF-8", "JavaScript", and the date "30/10/2023".

2. 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:

```
var theObject=JSON.parse(message.body);
```



- Modificamos en app.js la función connectAndSubscribe

```
var theObject=JSON.parse(message.body);
```

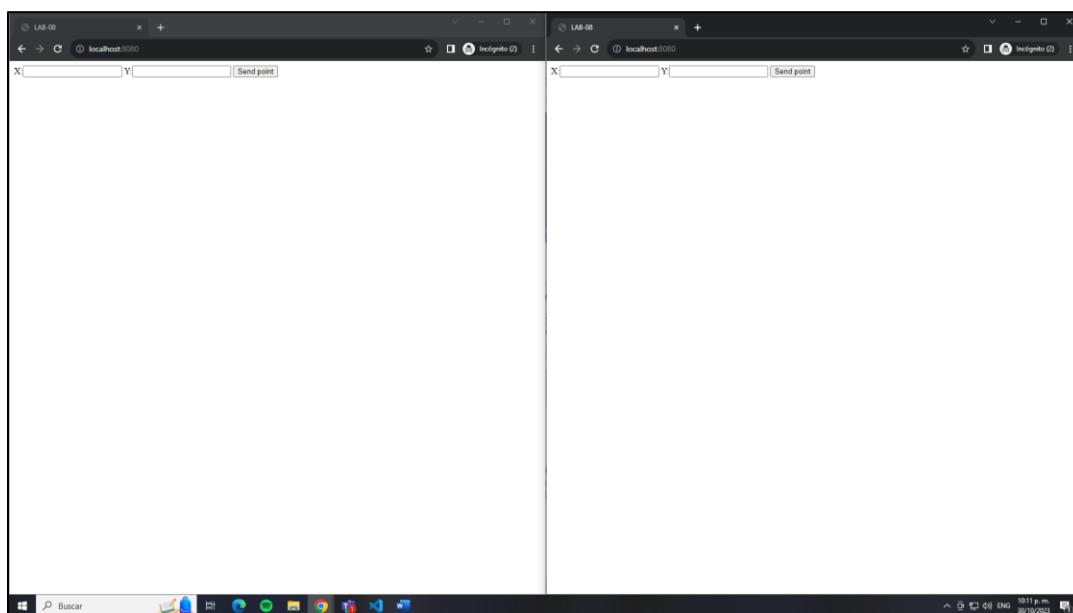
```
var connectAndSubscribe = function () {
    console.info('Connecting to WS...');
    var socket = new SockJS('/stompEndpoint');
    stompClient = Stomp.over(socket);

    //subscribe to /topic/TOPICXX when connections succeed
    stompClient.connect({}, function (frame) {
        console.log('Connected:' + frame);
        stompClient.subscribe('/topic/newpoint', function (eventbody) {
            let jsonObj = JSON.parse(eventbody.body);
            alert("Cordenadas recibidas: " + jsonObj.x + ", " + jsonObj.y);
        });
    });

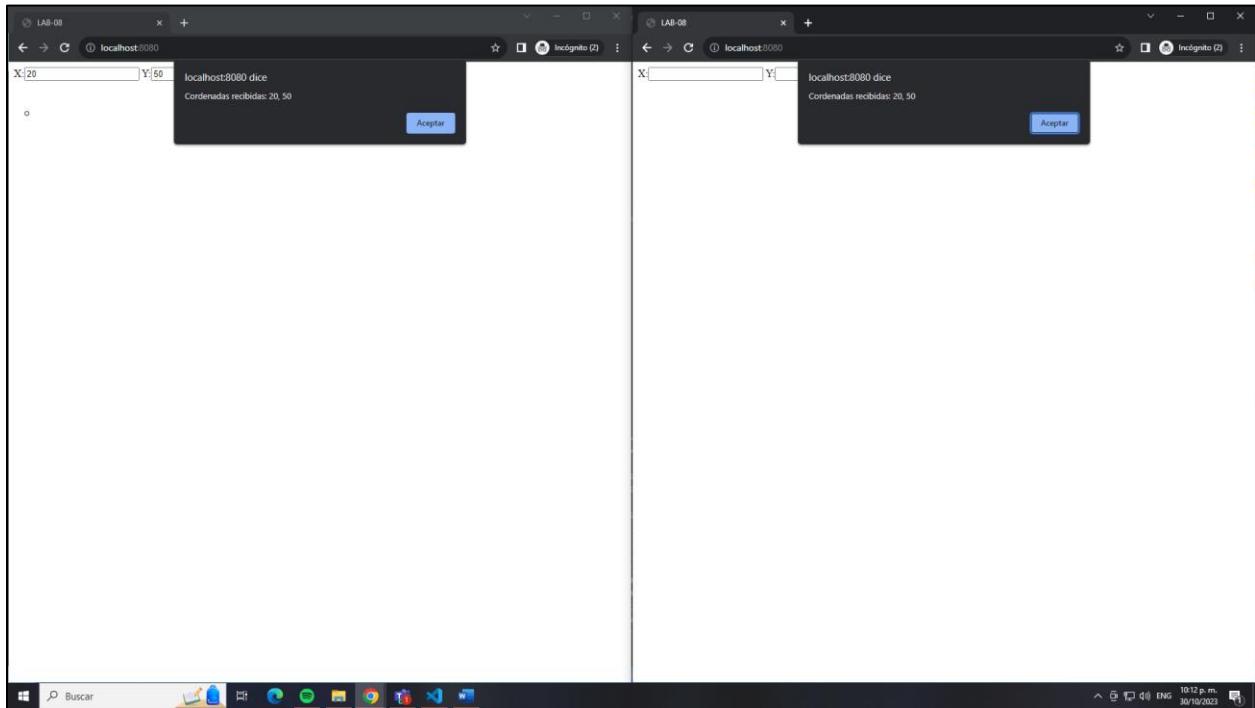
    return {
        init: function () {
            var can = document.getElementById("canvas");

            //websocket connection
            connectAndSubscribe();
        },
        publishPoint: function(px,py){
            var pt=new Point(px,py);
            console.info("publishing point at "+pt);
            addPointToCanvas(pt);
        }
    };
};
```

3. 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).



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



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

```
git commit -m "PARTE 1".
```



- 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.

```

var connectAndSubscribe = function () {
    console.info('Connecting to WS...');
    var socket = new SockJS('/stompEndpoint');
    stompClient = Stomp.over(socket);

    //subscribe to /topic/TOPICXX when connections succeed
    stompClient.connect({}, function (frame) {
        console.log('Connected: ' + frame);
        stompClient.subscribe('/topic/newpoint', function (eventbody) {
            let jsonObj = JSON.parse(eventbody.body);
            //alert("Coordenadas recibidas: " + jsonObj.x + " " + jsonObj.y);
            addPointToCanvas(new Point(jsonObj.x, jsonObj.y));
        });
    });
};

return;
}

```

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

```

init: function () {
    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");

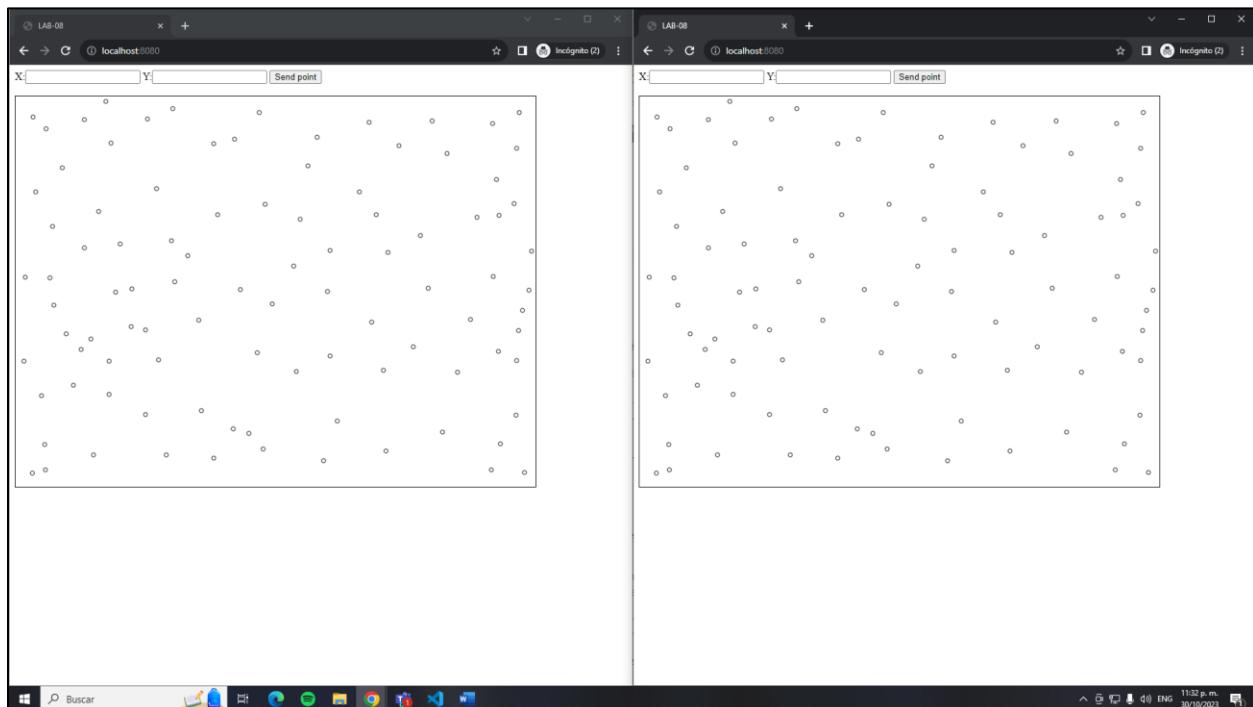
    if(window.PointerEvent) {
        canvas.addEventListener("pointerdown", function(event){
            let mousePosition = getMousePosition(event);
            app.publishPoint(mousePosition.x, mousePosition.y);
        });
    }else{
        canvas.addEventListener("mousedown", function(event){
            let mousePosition = getMousePosition(event);
            app.publishPoint(mousePosition.x, mousePosition.y);
        });
    }

    //websocket connection
    connectAndSubscribe();
},
publishPoint: function(px,py){
    var pt=new Point(px,py);
    console.info("publishing point at "+pt);
    //addPointToCanvas(pt);

    //publicar el evento
    //creando un objeto literal
    stompClient.send("/topic/newpoint", {}, JSON.stringify(pt));
},

```

- Ejecute su aplicación en varios navegadores (y si puede en varios computadores, accediendo a la aplicación mediante 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.



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

```
git commit -m "PARTE 2".
```



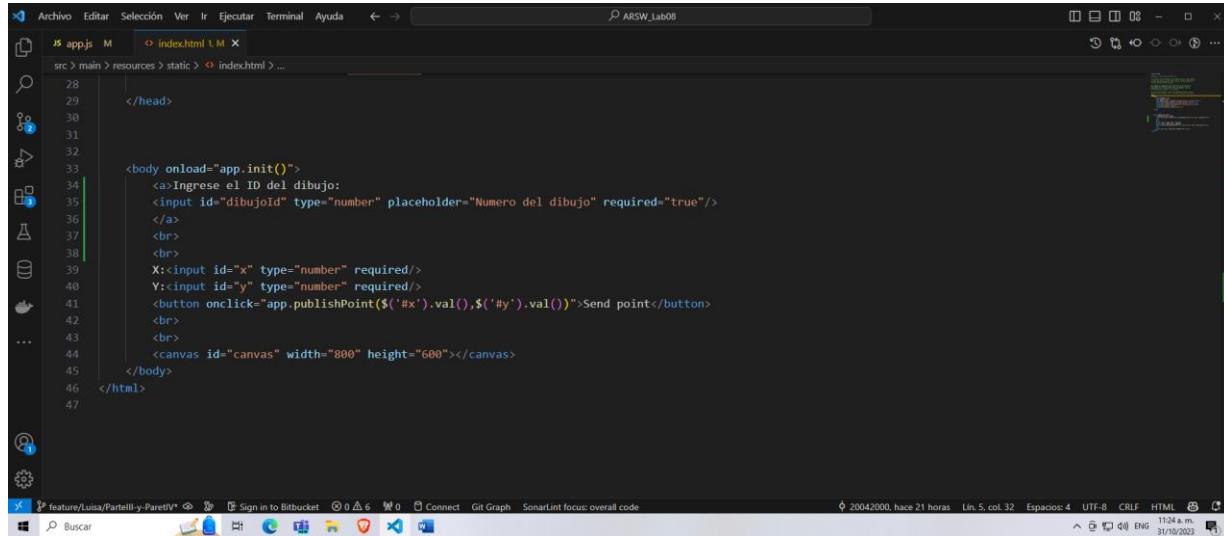
- Visualizar los commits del repositorio.

## 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.

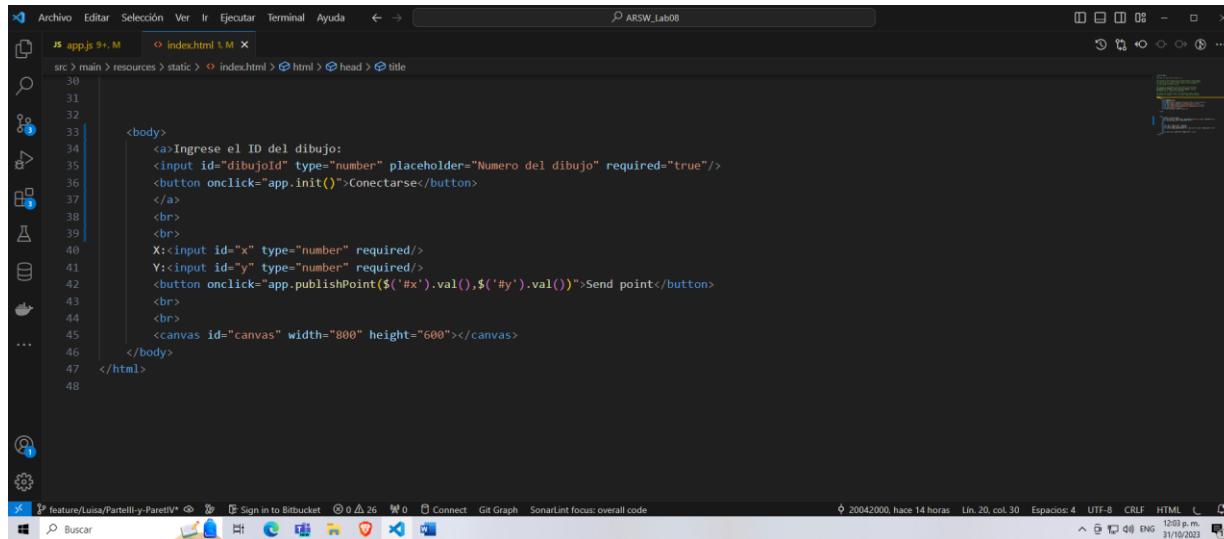


The screenshot shows the Visual Studio Code interface with the 'index.html' file open. The code has been modified to include a new input field for drawing ID:

```
<body onload="app.init()>
  <a>Ingrese el ID del dibujo:</a>
  <input id="dibujoid" type="number" placeholder="Número del dibujo" required/>
</body>
```

2. 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.



The screenshot shows the Visual Studio Code interface with the 'index.html' file open. A new button has been added to handle the connection logic:

```
<button onclick="app.init()">Conectarse</button>
```

## ➤ Modificamos app.js

```

1 var app = (function () {
2
3     class Point{
4         constructor(x,y){
5             this.x=x;
6             this.y=y;
7         }
8     }
9
10    var stompClient = null;
11    var dibujoId = null;
12
13    var addPointToCanvas = function (point) {
14        var canvas = document.getElementById("canvas");
15        var ctx = canvas.getContext("2d");
16        ctx.beginPath();
17        ctx.arc(point.x, point.y, 3, 0, 2 * Math.PI);
18        ctx.stroke();
19    };
20
21    var getMousePosition = function (evt) [
22        canvas = document.getElementById("canvas");
23    ];
24
25    var rect = canvas.getBoundingClientRect();
26    var x: evt.clientX - rect.left,
27    y: evt.clientY - rect.top
28];
29
30
31    var connectAndSubscribe = function (dibujoId) {
32        console.info('Connecting to WS...');
33        var socket = new SockJS('/stompEndpoint');
34        stompClient = Stomp.over(socket);
35
36        //subscribe to /topic/TOPICXX when connections succeed
37        stompClient.connect({}, function (frame) {
38            console.log('Connected: ' + frame);
39            stompClient.subscribe('/topic/newpoint' + dibujoId, function (event) {
40                let jsonObj = JSON.parse(event.body);
41                //alert("Coordenadas recibidas: " + jsonObj.x + ", " + jsonObj.y);
42                addPointToCanvas(new Point(jsonObj.x, jsonObj.y));
43
44            });
45        });
46    };
47
48};
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```

```

1 var rect = canvas.getBoundingClientRect();
2 var x: evt.clientX - rect.left,
3 y: evt.clientY - rect.top
4;
5
6    var connectAndSubscribe = function (dibujoId) {
7        console.info('Connecting to WS...');

8        var socket = new SockJS('/stompEndpoint');
9        stompClient = Stomp.over(socket);

10       //subscribe to /topic/TOPICXX when connections succeed
11       stompClient.connect({}, function (frame) {
12           console.log('Connected: ' + frame);
13           stompClient.subscribe('/topic/newpoint' + dibujoId, function (event) {
14               let jsonObj = JSON.parse(event.body);
15               //alert("Coordenadas recibidas: " + jsonObj.x + ", " + jsonObj.y);
16               addPointToCanvas(new Point(jsonObj.x, jsonObj.y));
17
18           });
19       });
20   };
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71

```

```
JS app.js 9+ M index.html 1 M
src > main > resources > static > JS app.js > [app] > <function> > [getMousePosition]
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80

JS app.js 9+ M index.html 1 M
src > main > resources > static > JS app.js > [app] > <function> > [getMousePosition]
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80

JS app.js 9+ M index.html 1 M
src > main > resources > static > JS app.js > [app] > <function> > [getMousePosition]
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
```

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

```
JS app.js 9+ M index.html 1 M
src > main > resources > static > JS app.js > [app] > <function> > [getMousePosition]
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105

JS app.js 9+ M index.html 1 M
src > main > resources > static > JS app.js > [app] > <function> > [getMousePosition]
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
```

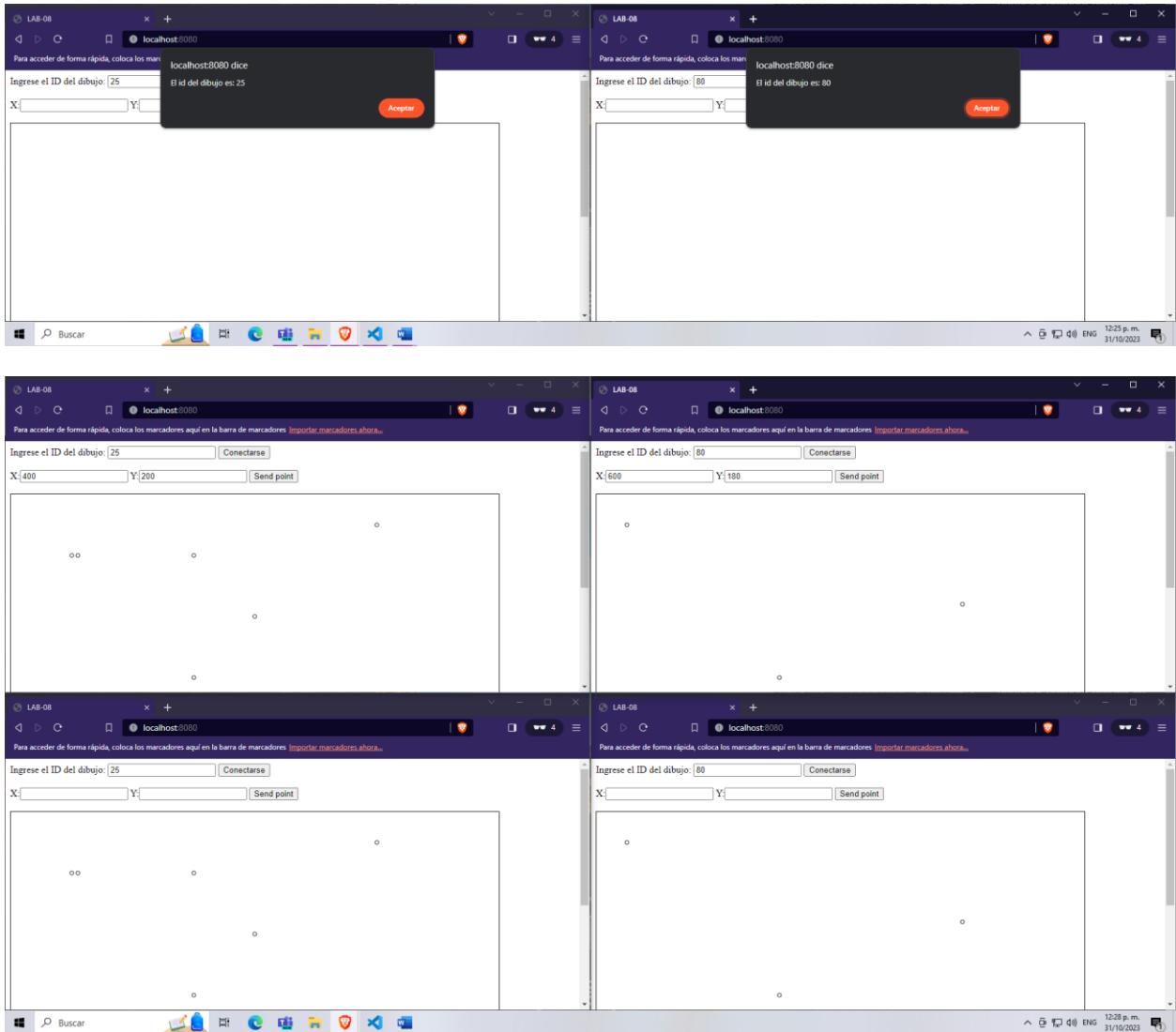
4. Rectifique que se puedan realizar dos dibujos de forma independiente, cada uno de éstos entre dos o más clientes.

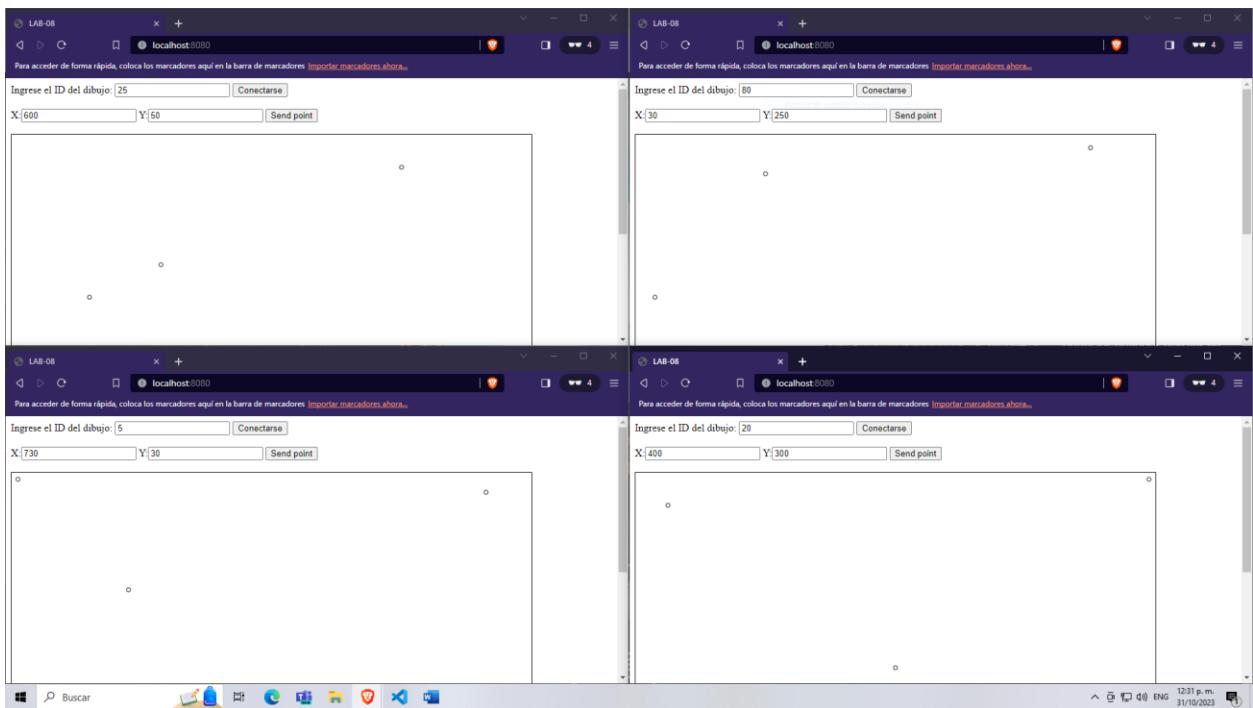
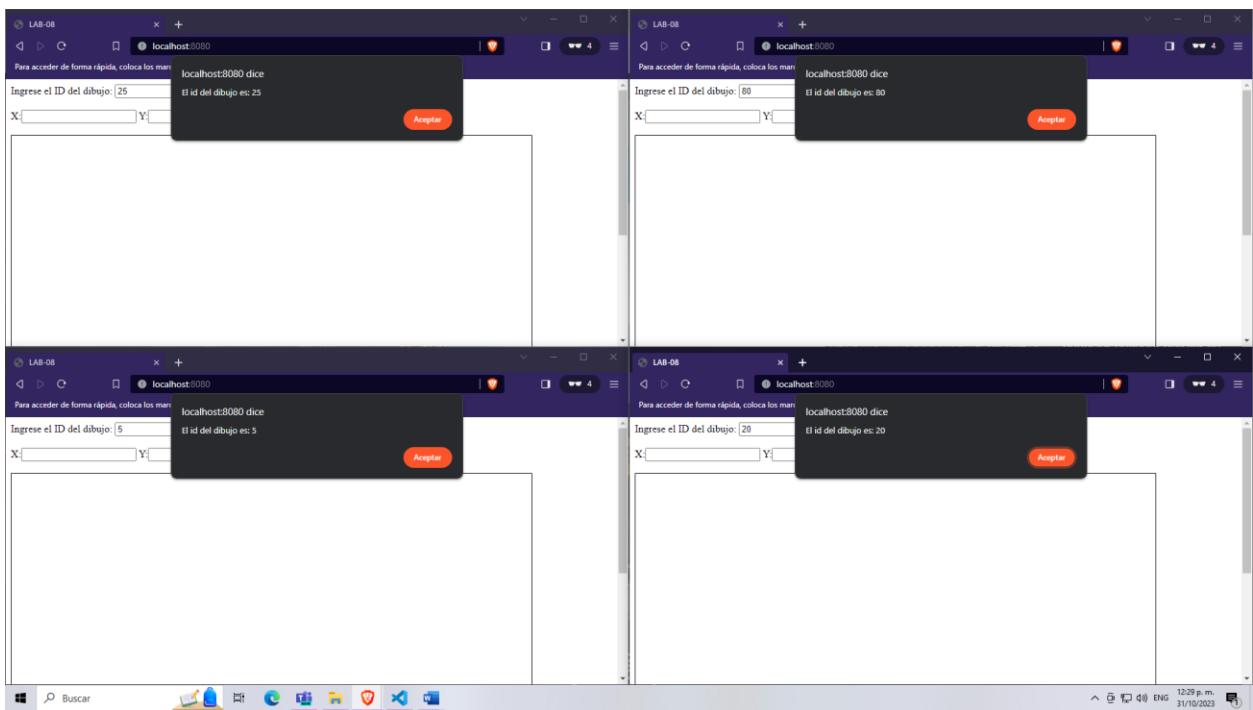
```
git commit -m "PARTE 3".
```



- Visualizar los commits del repositorio.

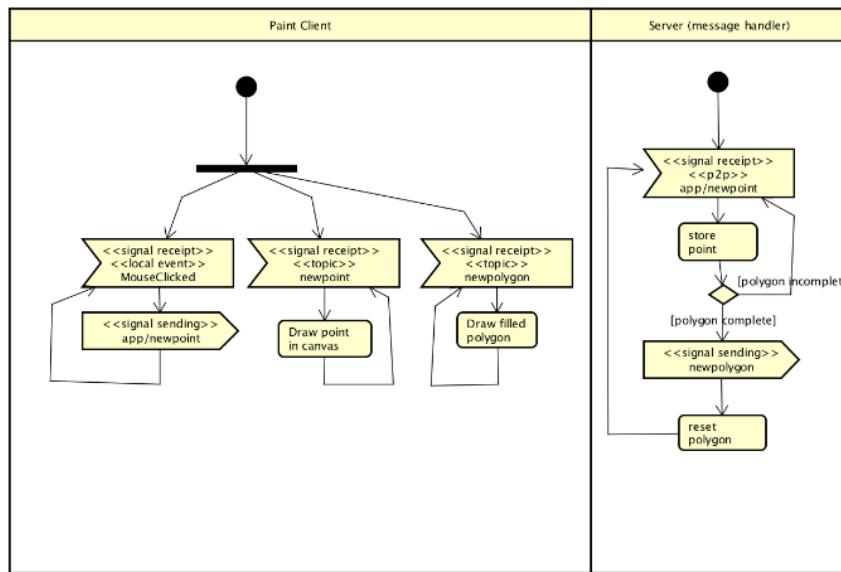
➤ Resultados ejecución:





## 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".

```
@Controller
public class STOMPMessagesHandler {

    @Autowired
    SimpMessagingTemplate msgt;

    @MessageMapping("/newpoint.{numdibujo}")
    public void handlePointEvent(Point pt,@DestinationVariable String numdibujo) throws Exception {
        System.out.println("Nuevo punto recibido en el servidor!:" + pt);
        msgt.convertAndSend("/topic/newpoint" + numdibujo, pt);
    }
}
```

- Se creo la clase StompMessagesHandler

The screenshot shows the Eclipse IDE interface with the following details:

- Left Sidebar (Explorador):** Shows the project structure for "ARSW\_Lab08". The "src" folder contains "main" and "resources" subfolders. "main" contains "java" (with "edu/uci/arsw/collabpaint/controller/StompMessagesHandler.java" selected), "model" (with "CollabPaintApplication.java" and "CollabPaintWebSocketConfig.java"), "static" (with "index.html" selected), and "resources" (with "application.properties"). Other visible files include "js/app.js" and "# styles.css".
- Central Area:** Displays two open tabs:
  - index.html:** A simple HTML file with the content: <h1>ARSW\_Lab08</h1>.
  - StompMessagesHandler.java [2 U]:** Java code for a Spring controller. It imports various Spring packages and defines a controller class "StompMessagesHandler" with a method "handlePointEvent" that prints a message to the console and sends it to a topic using a "SimpMessagingTemplate".
- Bottom Status Bar:** Shows the current file path ("feature/Luisa/PartelliY-ParettiV-1-parteI"), line number (Line 13), column number (col 15), and other status information like "Espacios: 4", "UTF-8", "CR/LF", and "Java".

2. 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.

- ## ➤ Modificamos app.js

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda ↺ ↻ ARSW\_Lab08

EXPLORADOR

EDTORES ABIERTOS

- index.html 1
- J StompMessagesHandler.java 2. U
- JS app.js 9+, M x

src > main > resources > static > JS app.js > (o) app > <function> > publishPoint

You hace 9 minutos | 2 authors (20042000 and others)

```
1 van.app = (function () {  
2  
3 20042000, hace 23 horas | 1 author (20042000)  
4 class Point(  
5   constructor(x,y){  
6     this.x=x;  
7     this.y=y;  
8   }  
9 }  
10  
11 var stompClient = null;  
12 //var dibujoId = null;  
13 var numdibujos = null;  
14  
15 var addPointToCanvas = function (point) {  
16   var canvas = document.getElementById("canvas");  
17   var ctx = canvas.getContext("2d");  
18   ctx.beginPath();  
19   ctx.arc(point.x, point.y, 3, 0, 2 * Math.PI);  
20   ctx.stroke();  
21 }  
22  
23 var get.mousePosition = function (evt) {  
24  
25 feature/Luisa/Partelli-y-ParettiV* Sign in to Bitbucket 0 29 Connect Git Graph SonarUnit focus: overall code You, hace 50 minutos Lin. 91, col. 10 Espacio: 4 UTF-8 CR LF JavaScript 31/10/2023
```

```
var connectAndSubscribe = function (numdibujo) {
    console.info('Connecting to WS...');
    var socket = new SockJS('/stompendpoint');
    stompClient = Stomp.over(socket);

    //subscribe to /topic/TOPICXX when connections succeed
    stompClient.connect({}, function (frame) {
        console.log('Connected: ' + frame);
        stompClient.subscribe('/topic/newpoint' + numdibujo, function (eventbody) {
            let jsonObj = JSON.parse(eventbody.body);
            //alert("Cordenadas recibidas: " + jsonObj.x + ", " + jsonObj.y);
            addPointToCanvas(new Point(jsonObj.x, jsonObj.y));
        });
    });

    return [
        init: function () {

```

```
init: function () {
    var can = document.getElementById("canvas");
    var context = canvas.getContext("2d");

    if(window.PointerEvent) {
        canvas.addEventListener("pointerdown", function(event){
            let.mousePosition = getMousePosition(event);
            app.publishPoint(mousePosition.x, mousePosition.y);
        });
    }else{
        canvas.addEventListener("mousedown", function(event){
            let.mousePosition = getMousePosition(event);
            app.publishPoint(mousePosition.x, mousePosition.y);
        });
    }

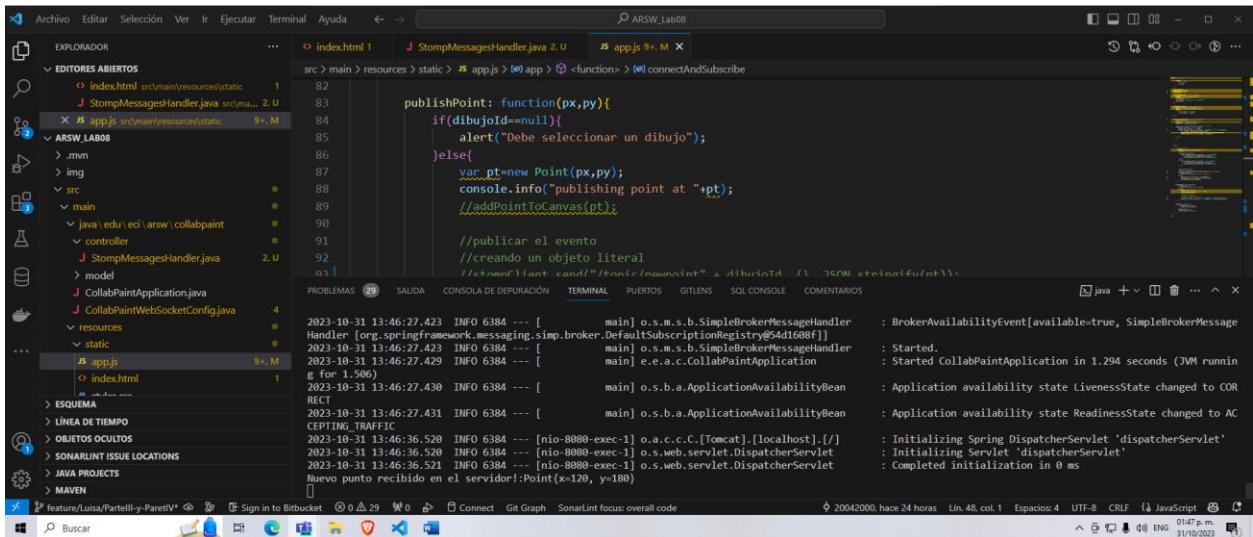
    //websocket connection
    //connectAndSubscribe();
    numdibujo = parseInt(document.getElementById("dibujoid").value);
    if(isNaN(numdibujo)){
        alert("El id del dibujo debe ser un numero");
    }else{
        alert("El id del dibujo es: " + numdibujo);
        can.getContext("2d").clearRect(0, 0, 800, 600);
        connectAndSubscribe(numdibujo);
    }
},

```

```
publishPoint: function(px,py){
    if(dibujoid==null){
        alert("Debe seleccionar un dibujo");
    }else{
        var pt=new Point(px,py);
        console.info("publishing point at "+pt);
        //addPointToCanvas(pt);

        //publicar el evento
        //creando un objeto literal
        //stompClient.send("/topic/newpoint" + dibujoid, {}, JSON.stringify(pt));
        stompClient.send("/app/newpoint." + numdibujo, {}, JSON.stringify(pt));
    }
},
disconnect: function () {
    if (stompClient !== null) {
        stompClient.disconnect();
    }
    setConnected(false);
    console.log("Disconnected");
}
};
```

➤ Resultados ejecución:



The screenshot shows the VS Code interface with the ARSW\_Lab08 project open. The Explorer sidebar shows files like index.html, StompMessagesHandler.java, and app.js. The Terminal tab displays application logs and configuration details. The code editor shows a snippet of JavaScript for publishing points to a canvas.

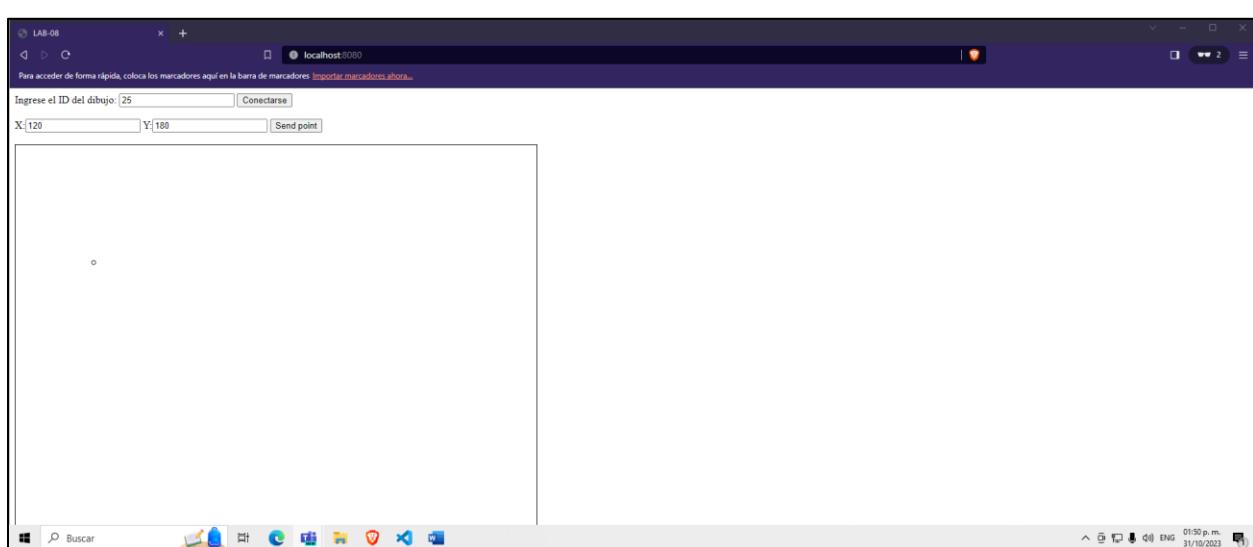
```

    publishPoint: function(px,py){
        if(dibujoId==null){
            alert("Debe seleccionar un dibujo");
        }else{
            var pt=new Point(px,py);
            console.info("publishing point at "+pt);
            //addPointToCanvas(pt);
        }
    }
}

```

The terminal output includes log entries from the application and Spring framework, such as:

- INFO [main] o.s.m.b.SimpleBrokerMessageHandler : BrokerAvailabilityEvent[available=true, SimpleBrokerMessageHandler [org.springframework.messaging.simp.broker.DefaultSubscriptionRegistry@5d1608f]]
- INFO [main] o.s.m.b.SimpleBrokerMessageHandler : Started.
- INFO [main] e.e.a.CollabPaintApplication : Started CollabPaintApplication in 1.294 seconds (JVM running for 1.506)
- INFO [main] o.s.b.a.ApplicationAvailabilityBean : Application availability state LivenessState changed to CORRECTING\_TRAFFIC
- INFO [main] o.s.b.a.ApplicationAvailabilityBean : Application availability state ReadinessState changed to ACTIVE
- INFO [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
- INFO [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
- INFO [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed Initialization in 0 ms

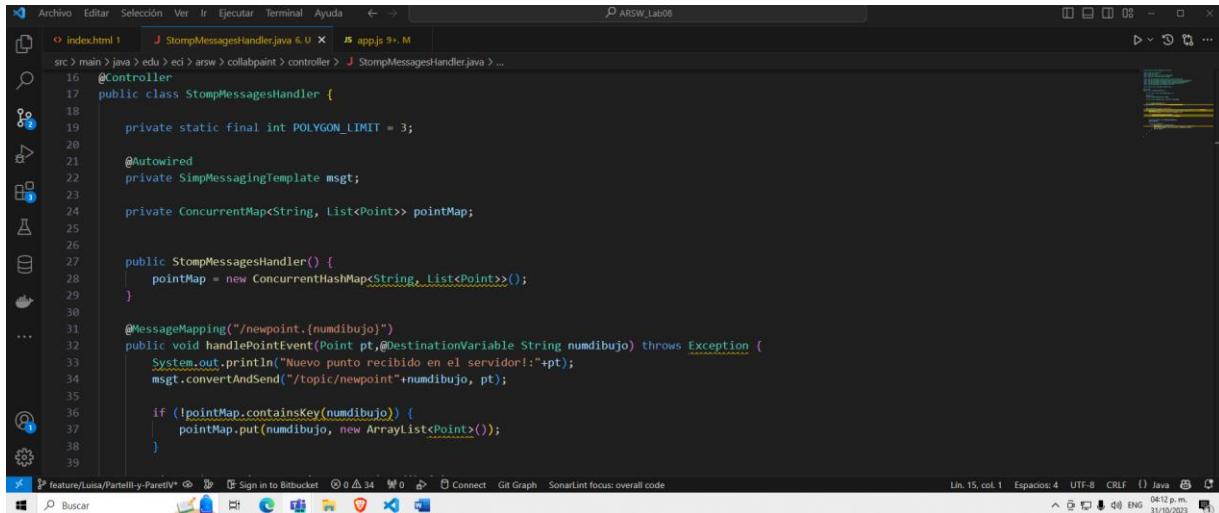
**LAB-08** 

The browser window shows a canvas with a single point at coordinates (120, 180). The URL is localhost:8080.

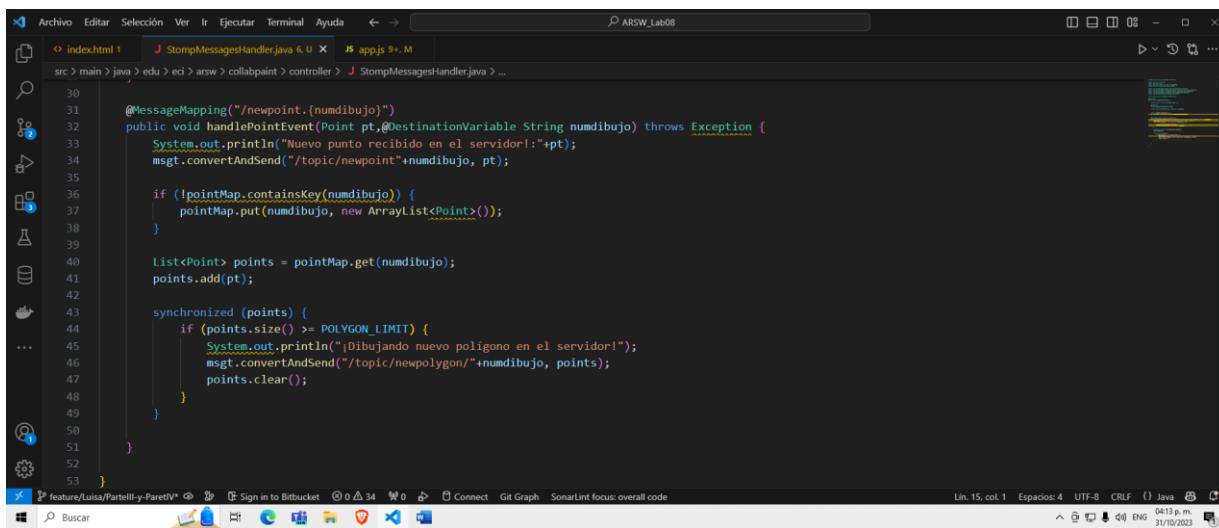
3. Una vez rectificado el funcionamiento, se quiere aprovechar este 'interceptor' de eventos para cambiar ligeramente la funcionalidad:
  - i. Se va a manejar un nuevo tópico llamado '/topic/newpolygon.{numdibujo}', en donde el lugar de puntos, se recibirán objetos javascript que tengan como propiedad un conjunto de puntos.
  - ii. 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!

- Modificamos la clase StompMessagesHandler



```
16 @Controller
17 public class StompMessagesHandler {
18     private static final int POLYGON_LIMIT = 3;
19
20     @Autowired
21     private SimpMessagingTemplate msgt;
22
23     private ConcurrentMap<String, List<Point>> pointMap;
24
25
26     public StompMessagesHandler() {
27         pointMap = new ConcurrentHashMap<String, List<Point>>();
28     }
29
30     @MessageMapping("/newpoint.{numdibujo}")
31     public void handlePointEvent(Point pt,@DestinationVariable String numdibujo) throws Exception {
32         System.out.println("Nuevo punto recibido en el servidor!:"+pt);
33         msgt.convertAndSend("/topic/newpoint"+numdibujo, pt);
34
35         if (!pointMap.containsKey(numdibujo)) {
36             pointMap.put(numdibujo, new ArrayList<Point>());
37         }
38
39     }
40 }
```



```
30     @MessageMapping("/newpoint.{numdibujo}")
31     public void handlePointEvent(Point pt,@DestinationVariable String numdibujo) throws Exception {
32         System.out.println("Nuevo punto recibido en el servidor!:"+pt);
33         msgt.convertAndSend("/topic/newpoint"+numdibujo, pt);
34
35         if (!pointMap.containsKey(numdibujo)) {
36             pointMap.put(numdibujo, new ArrayList<Point>());
37         }
38
39         List<Point> points = pointMap.get(numdibujo);
40         points.add(pt);
41
42         synchronized (points) {
43             if (points.size() >= POLYGON_LIMIT) {
44                 System.out.println("Dibujando nuevo polígono en el servidor!");
45                 msgt.convertAndSend("/topic/newpolygon"+numdibujo, points);
46                 points.clear();
47             }
48         }
49     }
50 }
51 }
```

- iii. 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.

- Modificamos app.js agregando la función addPolygonToCanva y agregamos la suscripción del tópico '/topic/newpolygon'.



```
src > main > resources > static > app.js > app > <function> > init
  19   CLX.S.LTROKE();
  20 }
  21
  22 var addPolygonToCanvas = function (points) {
  23   let c2 = canvas.getContext('2d');
  24   let init = false;
  25
  26   c2.fillStyle = '#f00';
  27   c2.beginPath();
  28   points.map(function (value, index) {
  29     if (init){
  30       c2.moveTo(value.x,value.y);
  31       init = true;
  32     } else {
  33       c2.lineTo(value.x,value.y);
  34     }
  35   });
  36   c2.closePath();
  37   c2.fill();
  38 };
  39
  40 var get.mousePosition = function (evt) {
  41   canvas = document.getElementById("canvas");
  42   var rect = canvas.getBoundingClientRect();
  43   return {
  44     x: evt.clientX - rect.left,
  45     y: evt.clientY - rect.top
  46   };
  47 };
  48
  49
  50
  51
  52
  53
  54
  55
  56
  57
  58
  59
  60
  61
  62
  63
  64
  65
  66
  67
  68
  69
  70
  71
  72
  73
  74
  75
  76
  77
  78
  79
  80
  81
  82
  83
  84
  85
  86
  87
  88
  89
  90
  91
  92
  93
  94
  95
  96
  97
  98
  99
  100
  101
  102
  103
  104
  105
  106
  107
  108
  109
  110
  111
  112
  113
  114
  115
  116
  117
  118
  119
  120
  121
  122
  123
  124
  125
  126
  127
  128
  129
  130
  131
  132
  133
  134
  135
  136
  137
  138
  139
  140
  141
  142
  143
  144
  145
  146
  147
  148
  149
  150
  151
  152
  153
  154
  155
  156
  157
  158
  159
  160
  161
  162
  163
  164
  165
  166
  167
  168
  169
  170
  171
  172
  173
  174
  175
  176
  177
  178
  179
  180
  181
  182
  183
  184
  185
  186
  187
  188
  189
  190
  191
  192
  193
  194
  195
  196
  197
  198
  199
  200
  201
  202
  203
  204
  205
  206
  207
  208
  209
  210
  211
  212
  213
  214
  215
  216
  217
  218
  219
  220
  221
  222
  223
  224
  225
  226
  227
  228
  229
  230
  231
  232
  233
  234
  235
  236
  237
  238
  239
  240
  241
  242
  243
  244
  245
  246
  247
  248
  249
  250
  251
  252
  253
  254
  255
  256
  257
  258
  259
  260
  261
  262
  263
  264
  265
  266
  267
  268
  269
  270
  271
  272
  273
  274
  275
  276
  277
  278
  279
  280
  281
  282
  283
  284
  285
  286
  287
  288
  289
  290
  291
  292
  293
  294
  295
  296
  297
  298
  299
  300
  301
  302
  303
  304
  305
  306
  307
  308
  309
  310
  311
  312
  313
  314
  315
  316
  317
  318
  319
  320
  321
  322
  323
  324
  325
  326
  327
  328
  329
  330
  331
  332
  333
  334
  335
  336
  337
  338
  339
  340
  341
  342
  343
  344
  345
  346
  347
  348
  349
  350
  351
  352
  353
  354
  355
  356
  357
  358
  359
  360
  361
  362
  363
  364
  365
  366
  367
  368
  369
  370
  371
  372
  373
  374
  375
  376
  377
  378
  379
  380
  381
  382
  383
  384
  385
  386
  387
  388
  389
  390
  391
  392
  393
  394
  395
  396
  397
  398
  399
  400
  401
  402
  403
  404
  405
  406
  407
  408
  409
  410
  411
  412
  413
  414
  415
  416
  417
  418
  419
  420
  421
  422
  423
  424
  425
  426
  427
  428
  429
  430
  431
  432
  433
  434
  435
  436
  437
  438
  439
  440
  441
  442
  443
  444
  445
  446
  447
  448
  449
  450
  451
  452
  453
  454
  455
  456
  457
  458
  459
  460
  461
  462
  463
  464
  465
  466
  467
  468
  469
  470
```

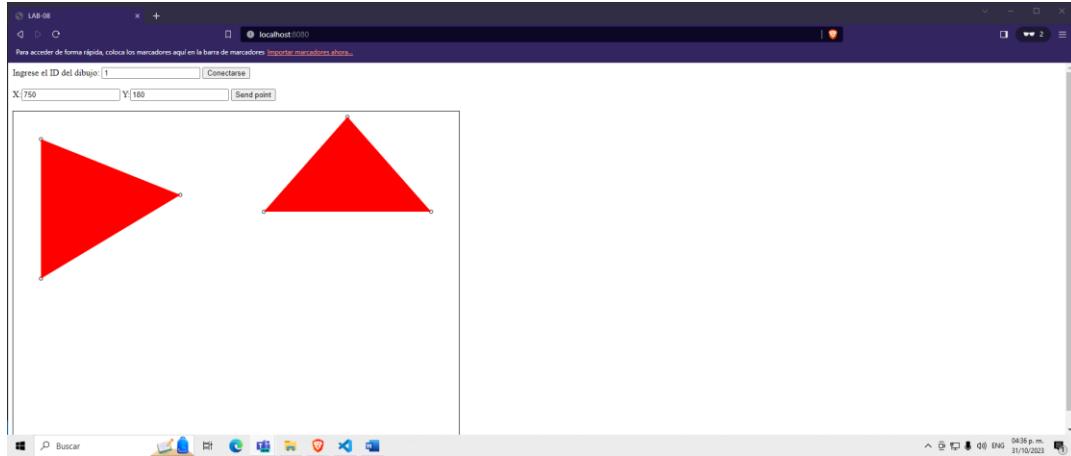
```
49 var connectAndSubscribe = function (numdibujo) {
50     console.info('Connecting to WS...');  
51     var socket = new SockJS('/stompendpoint');  
52     stompClient = Stomp.over(socket);  
53  
54     //subscribe to /topic/TOPICXX when connections succeed  
55     stompClient.connect({}, function (frame) {  
56         console.log("Connected: " + frame);  
57         stompClient.subscribe('/topic/newpoint' + numdibujo, function (eventbody) {  
58             let jsonObj = JSON.parse(eventbody.body);  
59             //alert("Cordenadas recibidas: " + jsonObj.x + ", " + jsonObj.y);  
60             addPointToCanvas(new Point(jsonObj.x, jsonObj.y));  
61         });  
62     });  
63     stompClient.subscribe('/topic/newpolygon/' + numdibujo, function (eventbody){  
64         addPolygonToCanvas(JSON.parse(eventbody.body));  
65     });  
66 };  
67 );  
68  
69 };  
70  
71  
72
```

- Resultados de ejecución para polígonos de 3 lados:

The screenshot shows the Eclipse IDE interface with several open tabs and windows. The main editor tab contains Java code for a `StompMessagesHandler`. The code includes a method `addPolygonToCanvas` that takes an array of points and adds them to a canvas context. The code is annotated with Javadoc and includes a note about uncommitted changes. Below the editor is a `PROBLEMS` view showing no issues. To the right, there is a `LOG` view displaying a log of events from a broker application, including subscription registrations and message exchanges. A `CONSOLA DE DEPURACIÓN` (Debug Console) tab is also visible.

```
src/main/resources/static/js/app.js (app <function> &gt; addPolygonToCanvas
  19
  20  );
  21
  22  var addPolygonToCanvas = function (points) {
  23      let c2 = canvas.getContext('2d');
  24      let init = false;
}
mp.broker.DefaultSubscriptionRegistry@469390ef]
2023-10-31 16:29:44.533 INFO 18184 --- [           main] o.s.m.s.b.stomp.BrokerMessageHandler : Started CollabPaintApplication in 1.321 seconds (JVM running for 1.54)
2023-10-31 16:29:44.533 INFO 18184 --- [           main] o.s.b.a.ApplicationAvailabilityBean : Application availability state LivenessState changed to CORRECT
2023-10-31 16:29:44.534 INFO 18184 --- [           main] o.s.b.a.ApplicationAvailabilityBean : Application availability state ReadinessState changed to ACCEPTING_TRAFFIC
2023-10-31 16:29:30.888 INFO 18184 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-10-31 16:29:30.888 INFO 18184 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : : Initializing Servlet 'dispatcherServlet'
2023-10-31 16:29:30.889 INFO 18184 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : : Completed initialization in 1 ms
Nuevo punto recibido en el servidor!|Point{x=50, y=50}
2023-10-31 16:29:44.647 INFO 18184 --- [MessageBroker-2] o.w.s.w.s.c.WebSocketMessageBrokerStats : WebSocketSession[1 current WS(1)-HttpStream(0)-HttpPull(0), 1 total, 0 closed abnormally (0 connect failures, 0 send limit, 0 transport errors)], stampRedisProtocolProcessed CONNEDT(1)-CONNECTED(1)-DISCONNECT(0), stampBrokerRelay(null), inboundChannel[pool size = 12, active threads = 0, queued tasks = 0, completed tasks = 12], outboundChannel[pool size = 2, active threads = 0, queued tasks = 0, completed tasks = 2, completed tasks = 3]
Nuevo punto recibido en el servidor!|Point{x=50, y=300}
Nuevo punto recibido en el servidor!|Point{x=300, y=-150}
¡Dibujando nuevo polígono en el servidor!
Nuevo punto recibido en el servidor!|Point{x=600, y=10}
Nuevo punto recibido en el servidor!|Point{x=450, y=-180}
Nuevo punto recibido en el servidor!|Point{x=750, y=-180}
¡Dibujando nuevo polígono en el servidor!
```

Nota: se puede pintar ingresando las coordenadas en los campos o pintando los puntos con el mouse al dar click sobre el canvas.



- iv. Verifique la funcionalidad: igual a la anterior, pero ahora dibujando polígonos cada vez que se agreguen cuatro puntos.

```
Archievador Editar Selección Ver Ir Ejecutar Terminal Ayuda <- > ARSW_Lab08

src > main > java > edu > eci > arsw > collabpaint > controller > J StompMessagesHandler.java > StompMessagesHandler > POLYGON_LIMIT

14 import edu.eci.arsw.collabpaint.model.Point;
15
16 @Controller
17 public class StompMessagesHandler {
18
19     private static final int POLYGON_LIMIT = 4;
20
21     @Autowired
22     private SimpMessagingTemplate msgt;
23
24     private ConcurrentMap<String, List<Point>> pointMap;
25
26
27     public StompMessagesHandler() {
28         pointMap = new ConcurrentHashMap<String, List<Point>>();
29     }
30
31     @MessageMapping("/newpoint.{numdibujo}")
32     public void handlePointEvent(Point pt,@DestinationVariable String numdibujo) throws Exception {
33         System.out.println("Nuevo punto recibido en el servidor:" + pt);
34         msgt.convertAndSend("/topic/newpoint"+numdibujo, pt);
35
36         if (!pointMap.containsKey(numdibujo)) {
37             pointMap.put(numdibujo, new ArrayList<Point>());
38         }
39     }
40 }
```

A screenshot of an IDE (IntelliJ IDEA) showing Java code for a controller named "StompMessagesHandler". The code handles Stomp messages to draw polygons. It uses a ConcurrentHashMap to store points for each drawing. The "POLYGON\_LIMIT" constant is set to 4. The "handlePointEvent" method receives a Point object and a destination variable "numdibujo", prints the point to the console, and then sends it to a topic. If the drawing key does not exist in the map, it is added with an empty list.

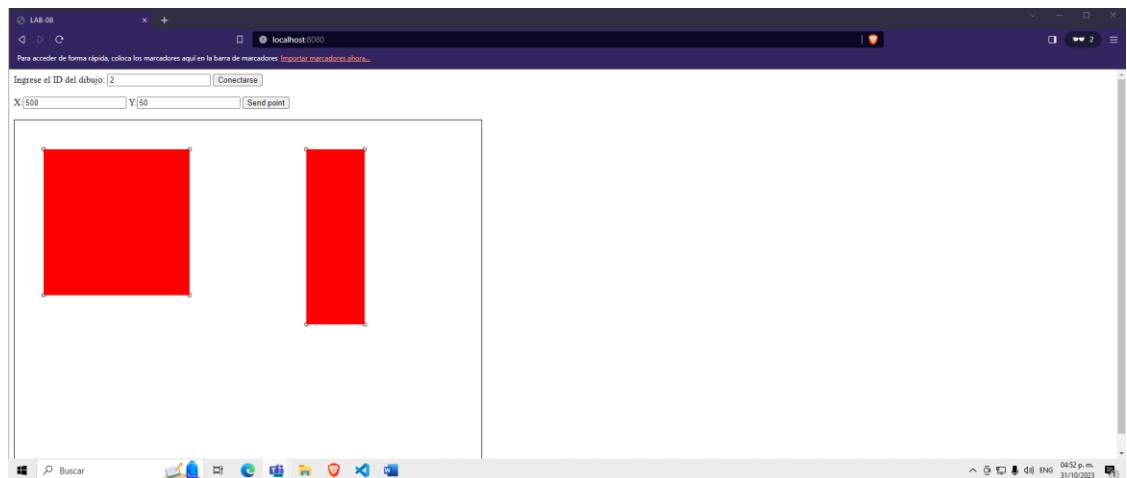
➤ Resultados de ejecución para polígonos de 4 lados:

```

index.html  x  J StompMessagesHandler.java 4.0  JS apps 9.0 M
src/main/java/edu/uci/arsw/collabpaint/controller/J StompMessagesHandler.java > StompMessagesHandler
14 import edu.uci.arsw.collabpaint.model.Point;
15
16 @Controller
17 public class StompMessagesHandler {
18
19     private static final int POLYGON_LIMIT = 4;
20
21     @Autowired
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS GITLENS SQL CONSOLE COMENTARIOS
2023-10-31 16:40:51.790 INFO 12208 --- [           main] o.s.b.a.ApplicationAvailabilityBean : Application availability state ReadinessState changed to ACCEPTING_TRAFFIC
2023-10-31 16:41:07.707 INFO 12208 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-10-31 16:41:07.707 INFO 12208 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2023-10-31 16:41:07.708 INFO 12208 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed Initialization in 0 ms
...
Nuevo punto recibido en el servidor! Point{x=50, y=50}
Nuevo punto recibido en el servidor! Point{x=50, y=50}
Nuevo punto recibido en el servidor! Point{x=300, y=300}
Nuevo punto recibido en el servidor! Point{x=300, y=300}
2023-10-31 16:41:51.494 INFO 12208 --- [MessageBroker-6] o.s.w.s.c.WebSocketMessageBrokerStats : WebSocketSession[1 current WS(1)-HttpStream(0)-HttpPoll(0), 1 total, 0 closed abnormally (0 connect failure, 0 send limit, 0 transport error)], stompSubProtocol[processed CONNECT((1)-CONNECTED(1)-DISCONNECT(0)), stompBrokerRelay=null], inboundChannel[pool size = 18, active threads = 0, queued tasks = 0, completed tasks = 18], outboundChannel[pool size = 4, active threads = 0, queued tasks = 0, completed tasks = 4], socketScheduler[pool size = 12, active threads = 1, queued tasks = 2, completed tasks = 11]
Nuevo punto recibido en el servidor! Point{x=300, y=50}
[Dibujando nuevo polígono en el servidor]
Nuevo punto recibido en el servidor! Point{x=600, y=50}
Nuevo punto recibido en el servidor! Point{x=600, y=350}
Nuevo punto recibido en el servidor! Point{x=500, y=350}
Nuevo punto recibido en el servidor! Point{x=500, y=50}
Nuevo punto recibido en el servidor! Point{x=500, y=50}
[Dibujando nuevo polígono en el servidor]
...

```

➤ Nota: se puede pintar ingresando las coordenadas en los campos o pintando los puntos con el mouse al dar click sobre el canvas.



4. 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.
5. Haga commit de lo realizado.

```
git commit -m "PARTE FINAL".
```



- Visualizar los commits del repositorio.

### **III. Conclusiones**

- Este laboratorio nos ha permitido comprender y aplicar conceptos clave de desarrollo web en tiempo real utilizando tecnologías como STOMP, WebSockets y HTML5 Canvas.
- Se aprovecharon las capacidades de WebSockets y STOMP para establecer una comunicación en tiempo real entre los clientes y el servidor. Esto ha permitido la colaboración instantánea en la creación de dibujos.