Blue Prints & Sockets

Juan Pablo Camargo Teherán

Santiago Alberto Naranjo Abril

Andres Felipe Parra Quiroga

Javier Ivan Toquica Barrera

Escuela Colombiana de Ingeniería Julio Garavito

1/29/2025

Introducción

En este proyecto se implementó una herramienta de dibujo colaborativo en la web, haciendo uso de WebSockets y el protocolo STOMP en un entorno Spring Boot. La aplicación permite que múltiples clientes se conecten simultáneamente, capturen eventos de interacción (ya sea mediante clicks o toques en dispositivos móviles) en un elemento HTML5 Canvas y compartan los puntos de dibujo en tiempo real. Para ello, se utilizaron tecnologías y conceptos como:

- HTML5, CSS3 y JavaScript: para construir la interfaz de usuario y manejar la lógica de captura y dibujo en el canvas.
- WebSockets y STOMP: que facilitan una comunicación bidireccional en tiempo real entre el cliente y el servidor, permitiendo la propagación inmediata de los eventos de dibujo a todas las instancias conectadas.
- Spring Boot: configurado como broker de mensajes, encargado de gestionar y distribuir los eventos de dibujo a través de tópicos, lo cual permite la escalabilidad y la gestión de múltiples sesiones de dibujo de manera concurrente.

El informe se enfoca principalmente en la segunda parte del proyecto, en la que se amplía la funcionalidad inicial para captar puntos mediante eventos en el canvas y replicar dichos eventos en múltiples clientes. Esto no solo involucra la transmisión y recepción de mensajes, sino también la integración de eventos de usuario y la manipulación gráfica en el lado del cliente. Además, se ha integrado la capacidad de gestionar múltiples sesiones de dibujo mediante tópicos dinámicos, lo que implica una profundización en el manejo de concurrencia y la sincronización en tiempo real.

Se evidencia la versatilidad y eficiencia de los WebSockets y el protocolo STOMP en el desarrollo de aplicaciones colaborativas. Se aprende a modularizar el código, a gestionar eventos complejos y a coordinar la comunicación entre el cliente y el servidor en aplicaciones distribuidas, lo cual es de gran utilidad en el desarrollo de sistemas interactivos y en tiempo real.

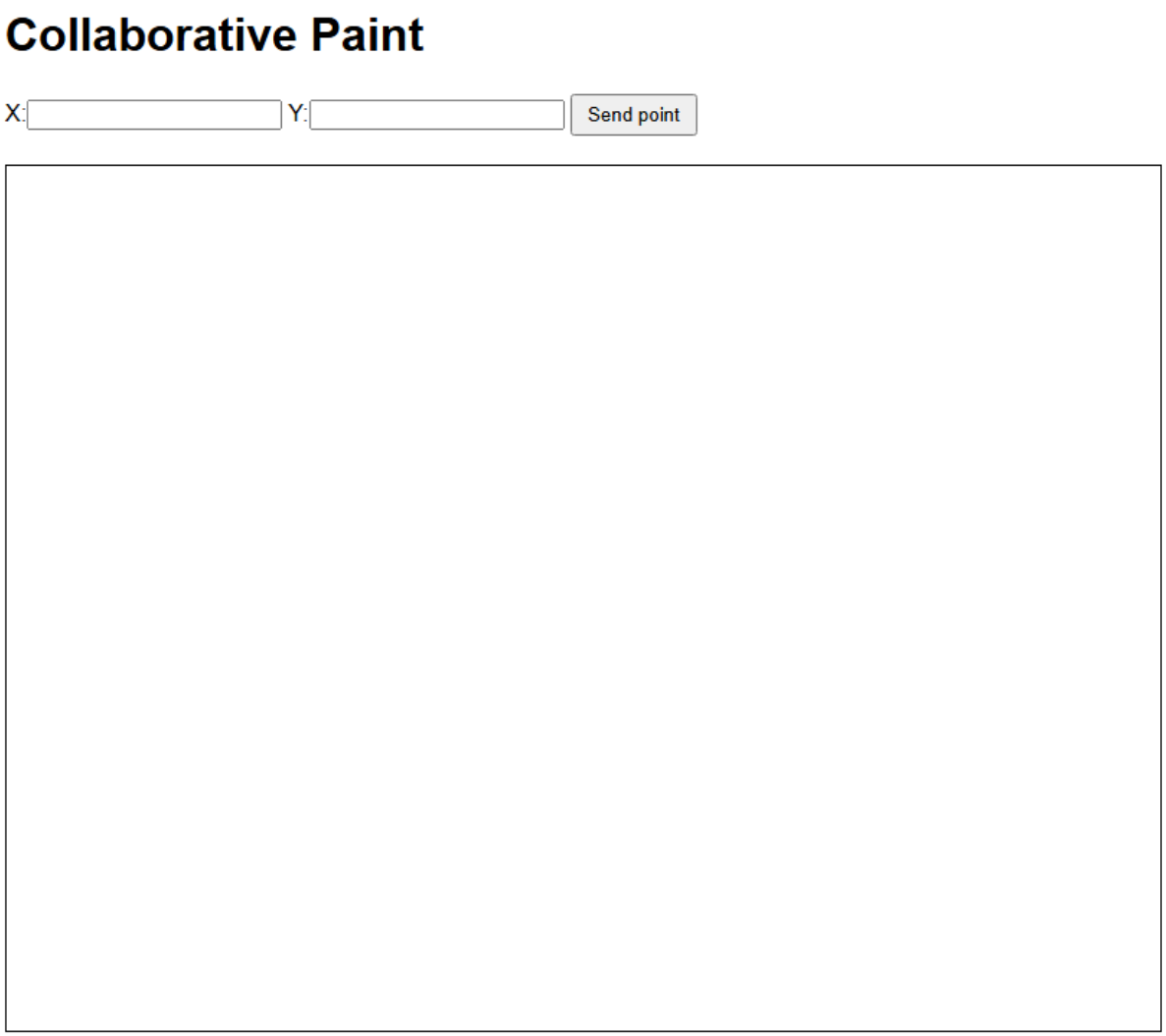
Desarrollo

Parte I: Configuración Básica de WebSockets

Objetivo: Establecer la conexión WebSocket y enviar/recepcionar puntos entre clientes.

Implementación:

1. **Configuración del servidor (CollabPaintWebSocketConfig)**
    a. Se habilitó el broker de mensajes STOMP con @EnableWebSocketMessageBroker.
    b. Se definió:
        i. **Prefijo para envío de mensajes:** /app
        ii. **Tópicos para suscripción:** /topic
        iii. **Endpoint WebSocket:** /stompendpoint (con soporte para SockJS).
2. **Manejo de mensajes (STOMPMessagesHandler)**
    a. Se creó un controlador con @MessageMapping("/newpoint") para recibir puntos.
    b. Cada punto recibido se reenvía a todos los clientes suscritos a /topic/newpoint usando SimpMessagingTemplate.
3. **Frontend (index.html y app.js)**
    a. Se configuró un cliente STOMP en JavaScript para conectarse al WebSocket.
    b. Se implementó:
        i. **Envío de puntos:** stompClient.send("/app/newpoint", {}, JSON.stringify(point))
        ii. **Recepción de puntos:** Suscripción a /topic/newpoint para dibujarlos en el canvas.



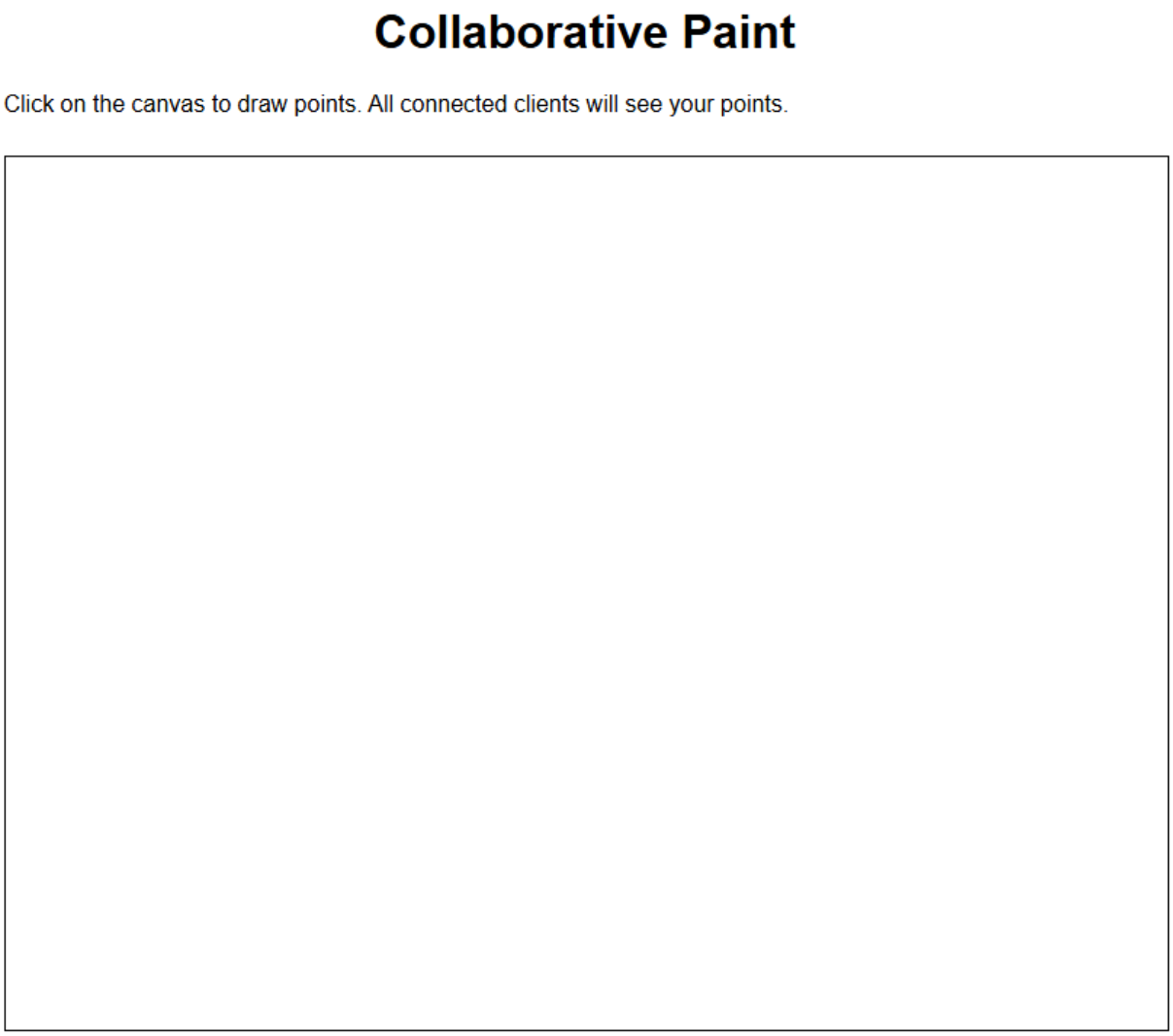
Parte II: Dibujo Colaborativo con Eventos de Canvas

Objetivo: Reemplazar la entrada manual de coordenadas por eventos de clic en el canvas.

Implementación:

1. **Captura de eventos en el canvas (app.js)**
    a. Se agregó un listener para click en el elemento <canvas>.
    b. Las coordenadas del clic se convierten en un objeto Point y se envían al servidor.
2. **Dibujo de puntos recibidos**

a. Se usó CanvasRenderingContext2D para dibujar círculos en las coordenadas recibidas.



Parte III: Soporte para Múltiples Dibujos Simultáneos

Objetivo: Permitir múltiples lienzos independientes usando tópicos dinámicos.

Implementación:

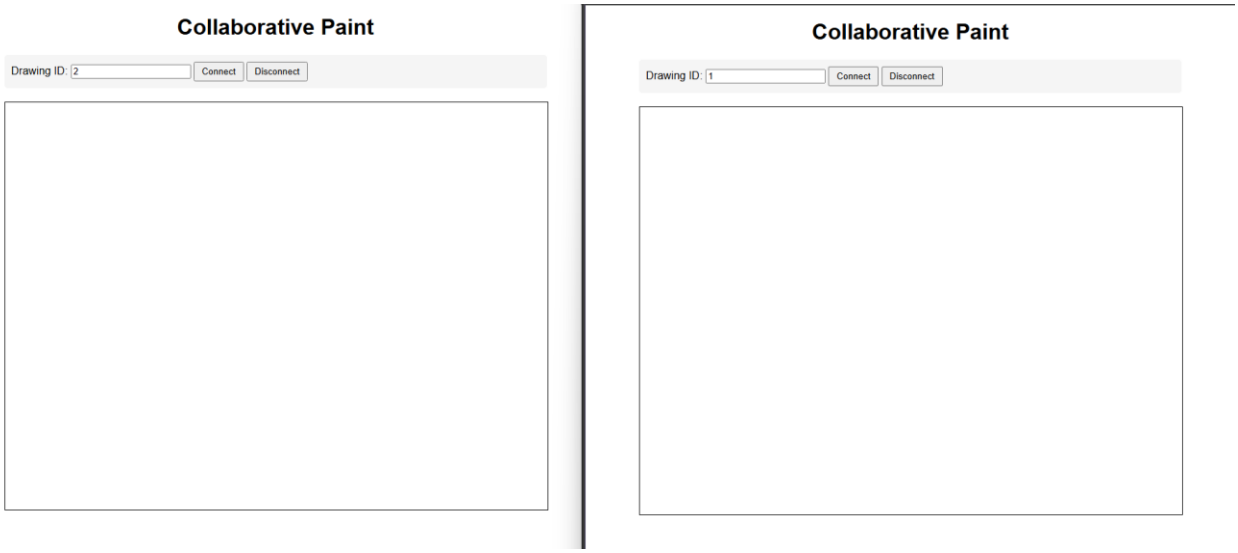
1. **Tópicos dinámicos (STOMPMessagesHandler)**
    a. Se modificó @MessageMapping para aceptar un ID de dibujo:

    @MessageMapping("/newpoint.{drawingId}")

b. Los mensajes se enviaban a tópicos específicos: /topic/newpoint.{drawingId}.

2. **Interfaz de usuario (index.html)**

   a. Se agregó un campo para ingresar el drawingId.



Parte IV: Dibujo Colaborativo de Polígonos

Objetivo: Detectar cuando se habían agregado 3+ puntos y dibujar polígonos.

Implementación:

1. **Almacenamiento de puntos en el servidor (STOMPMessagesHandler)**

   a. Se usó un ConcurrentHashMap<String, List<Point>> para guardar puntos por drawingId.

   b. Cuando un dibujo alcanzaba 3+ puntos, se enviaba la lista completa a /topic/newpolygon.{drawingId}.

2. **Dibujo de polígonos en el cliente (app.js)**

   a. Se suscribió a /topic/newpolygon.{drawingId}.

   b. Se implementó drawPolygon(points) para conectar los puntos con líneas usando CanvasRenderingContext2D.

Resultados y Conclusiones

La implementación de este proyecto permitió consolidar el uso de un broker de mensajes en Spring Boot junto a WebSockets y STOMP, facilitando una comunicación bidireccional de baja latencia crucial para aplicaciones colaborativas. La integración de HTML5 Canvas y el manejo de eventos de puntero lograron transformar entradas en representaciones gráficas en tiempo real, optimizando la experiencia interactiva. En conjunto, el proyecto cumplió con los requerimientos académicos y ofreció una experiencia práctica invaluable, demostrando el potencial de estas tecnologías para desarrollar soluciones de comunicación en tiempo real y la integración efectiva de sistemas distribuidos.

Bibliografías

STOMP. (n.d.). *STOMP: Simple (or Streaming) Text Oriented Messaging Protocol*. Recuperado el 28 de marzo de 2025, de https://stomp.github.io/

Spring Framework. (n.d.). WebSocket support in Spring. En Spring Framework Reference Documentation. Recuperado de https://docs.spring.io/spring-framework/docs/current/reference/html/web.html#websocket

MDN Web Docs. (n.d.). Canvas API. Recuperado de https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

MDN Web Docs. (n.d.). WebSocket API. Recuperado de https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

Chopra, V. (2013). WebSocket Essentials – Building Apps with HTML5 WebSockets. Packt Publishing.

Spring Boot. (n.d.). *Spring Boot Reference Documentation: WebSocket*. Recuperado de https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-websocket

Goetz, B., Peierls, T., Bloch, J., Bowbeer, J., Holmes, D., & Lea, D. (2006). *Java Concurrency in Practice*. Addison-Wesley Professional.

Wang, V., Salim, F., & Moskovits, P. (2011). *The Definitive Guide to HTML5 WebSocket*. Apress.