Característica	1. REST con Polling (Actual)	2. WebSocket (WS)	3. Server-Sent Events (SSE)
Complejidad Cliente (1-5)	1 (Muy Baja) Solo setInterval y fetch	5 (Alta) Manejo de estados, heartbeats, reconexión manual	2 (Baja) API nativa EventSource
Recursos Servidor (1k clientes)	Alto Desperdicio Miles de requests vacíos/minuto. Overhead de HTTP headers.	Eficiente 1 conexión abierta. Solo envía datos si cambian.	Eficiente. 1 conexión abierta. Similar a WS pero unidireccional.
Latencia (Ver cambios)	Alta (Promedio 2.5s) Depende del intervalo de polling.	Casi Nula (Real-time) Push inmediato.	Casi Nula (Real-time) Push inmediato.
Desconexión Breve	Robusto/Automático Falla un request, el siguiente a los 5s funciona.	Crítico La conexión muere. Debes programar la reconexión.	Resiliente El navegador intenta reconectar automáticamente.
Facilidad de Mockear	Muy Fácil Cualquier mock server REST funciona.	Difícil Necesitas un servidor WS dedicado o librerías complejas.	Media Requiere mockear un stream de texto text/event-stream.
Mi evaluacion	Debido al desperdicio de ciclos de CPU en el servidor esta opcion sera un caos una vez la operacion escale	Debido a que no hay operaciones en Real-time es un desproposito implementar websockets, tendria mas sentido en un chat o una plataforma de streaming	En mi opinion esta es la opcion correcta para este proyecto, eficiente, facil de mantener y cumple el proposito de responder a eventos