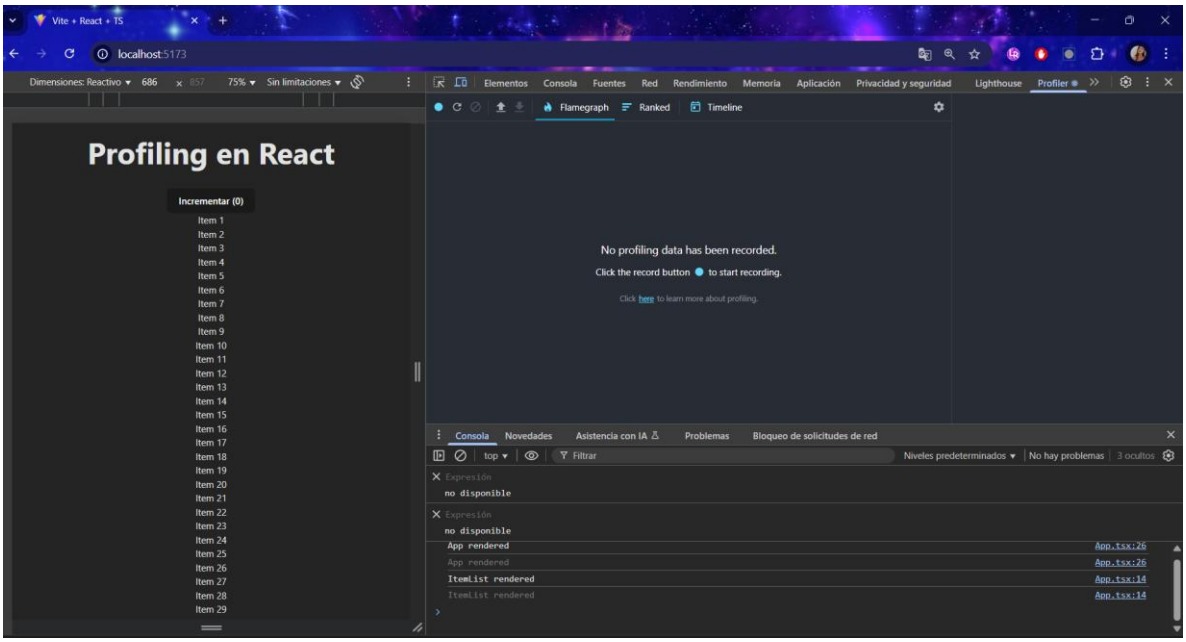
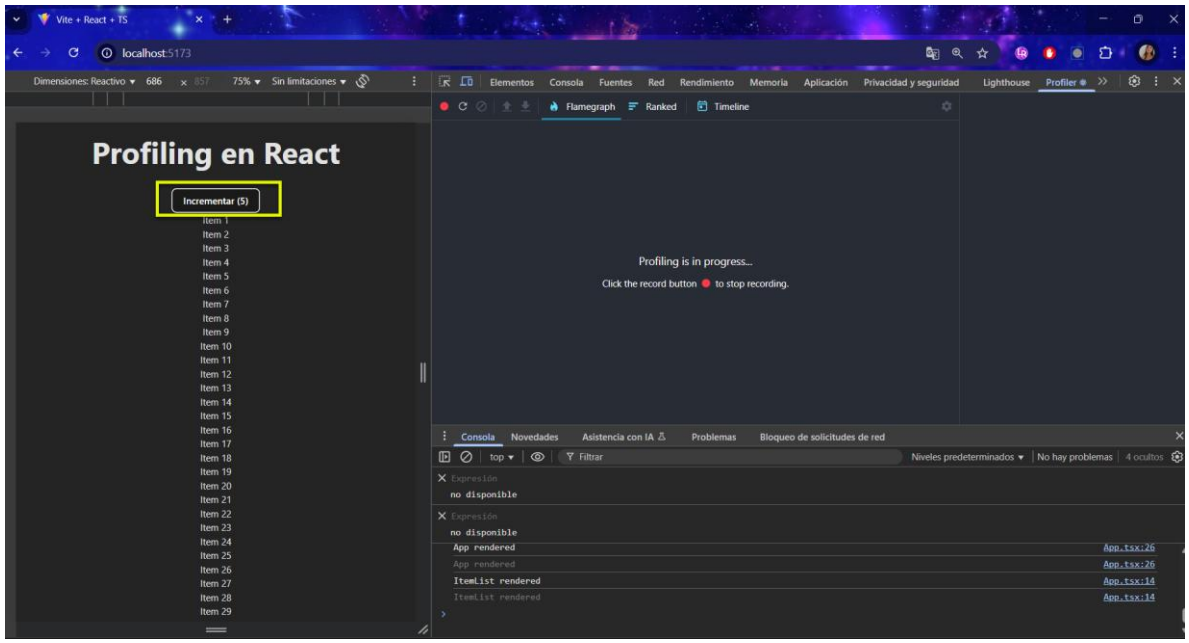
Capturas funcionamiento Profiler:

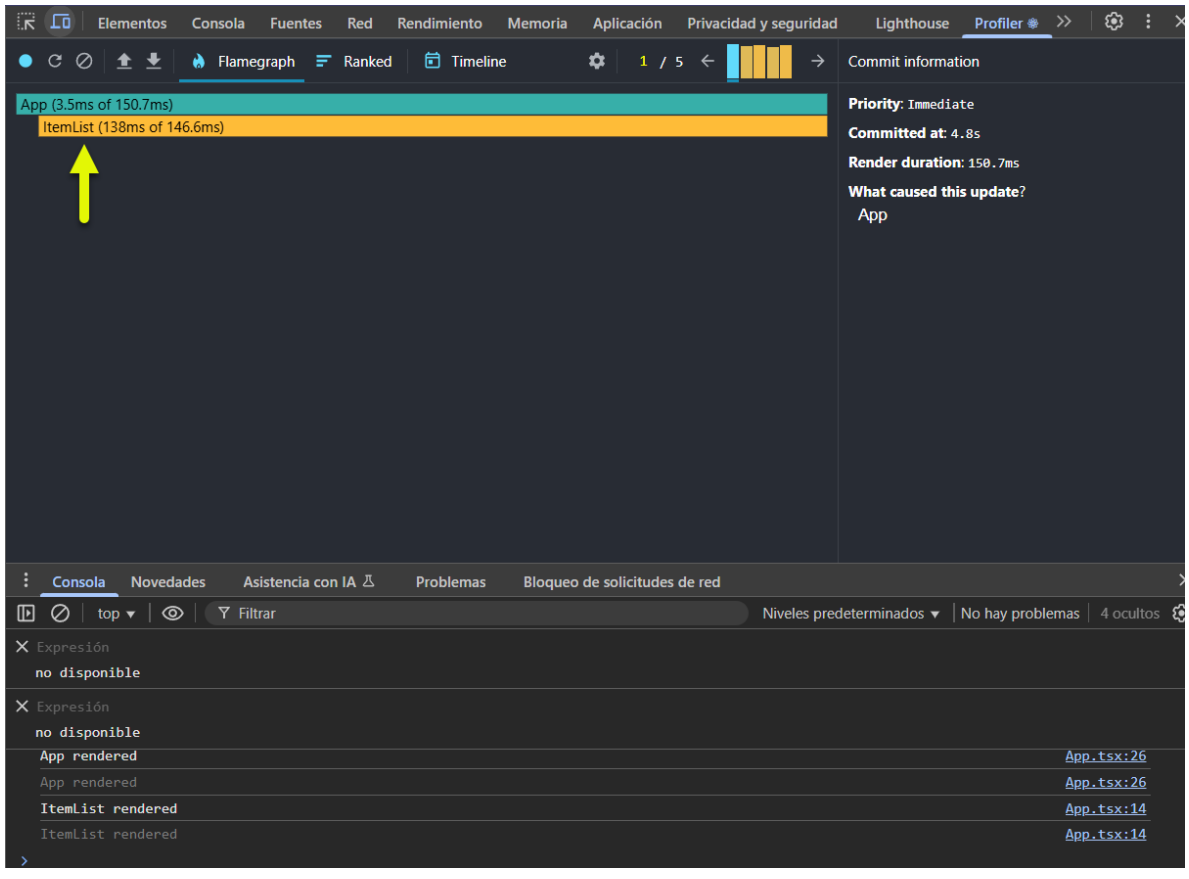
http://localhost:5173/

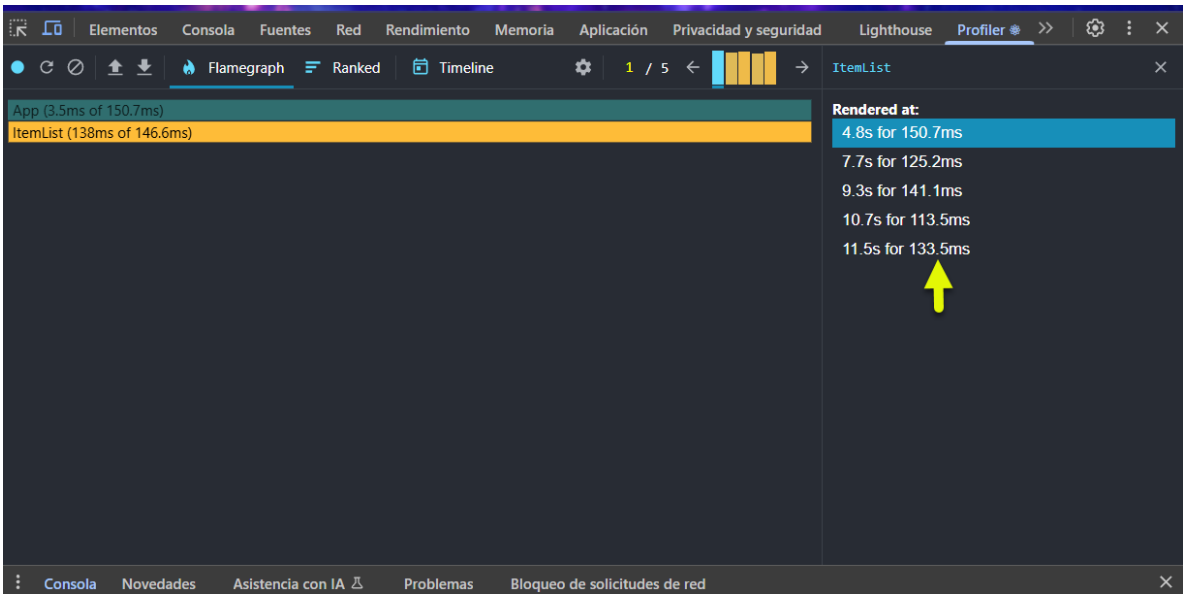
Ejecutamos el proyecto y con las Herramientas de Desarrollo del navegador, buscamos las pestañas "Components" y "Profiler".

Vamos a la pestaña de Profiler y observamos el comportamiento al hacer clic en el botón de "Incrementar" varias veces.




Vite + React + TS
localhost:5173
Profiling en React
Incrementar (5)
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 9
Item 10
Item 11
Item 12
Item 13
Item 14
Item 15
Item 16
Item 17
Item 18
Item 19
Item 20
Item 21
Item 22
Item 23
Item 24
Item 25
Item 26
Item 27
Item 28
Item 29
Elementos
Consola
Fuentes
Red
Rendimiento
Memoria
Aplicación
Privacidad y seguridad
Lighthouse
Profiler
Flamegraph
Ranked
Timeline
Profiling is in progress...
Click the record button to stop recording.
Consola
Novedades
Asistencia con IA
Problemas
Bloqueo de solicitudes de red
top
Filtrar
Niveles predeterminados
No hay problemas
4 ocultos
Expresión
no disponible
Expresión
no disponible
App rendered
App.tsx:26
App rendered
App.tsx:26
ItemList rendered
App.tsx:14
ItemList rendered
App.tsx:14


Elementos
Consola
Fuentes
Red
Rendimiento
Memoria
Aplicación
Privacidad y seguridad
Lighthouse
Profiler
Flamegraph
Ranked
Timeline
1 / 5
Commit information
App (3.5ms of 150.7ms)
ItemList (138ms of 146.6ms)
Priority: Immediate
Committed at: 4.8s
Render duration: 150.7ms
What caused this update?
App
Consola
Novedades
Asistencia con IA
Problemas
Bloqueo de solicitudes de red
top
Filtrar
Niveles predeterminados
No hay problemas
4 ocultos
Expresión
no disponible
Expresión
no disponible
App rendered
App.tsx:26
App rendered
App.tsx:26
ItemList rendered
App.tsx:14
ItemList rendered
App.tsx:14



Análisis:

En la imagen podemos observar uno de los commits o ciclos de renderizado, específicamente el primero.

En la parte derecha, bajo "Rendered at:", vemos una lista de los tiempos de renderizado. El que está resaltado (**4.8s for 150.7ms**).

Los otros tiempos (**7.7s for 125.2ms, etc**.) corresponden a los diferentes momentos en que la aplicación se actualizó (probablemente cada clic en "Incrementar").

Flamegraph:

- **App** (3.5ms of 150.7ms): Esta barra verde indica que el componente App se renderizó. Tomó 3.5ms de tiempo de "self-time" (tiempo que el propio componente tardó, sin contar a sus hijos) dentro de un renderizado total de 150.7ms para este commit.

- **ItemList** (138ms of 146.6ms): Esta barra, anidada debajo de App, está coloreada de amarillo. Esto es CRUCIAL. El color indica que el componente ItemList SÍ se renderizó durante este commit.

Tomó una cantidad significativa de tiempo: 138ms de "self-time" dentro de un renderizado total de 146.6ms (este tiempo incluye el renderizado de sus 1000 hijos).

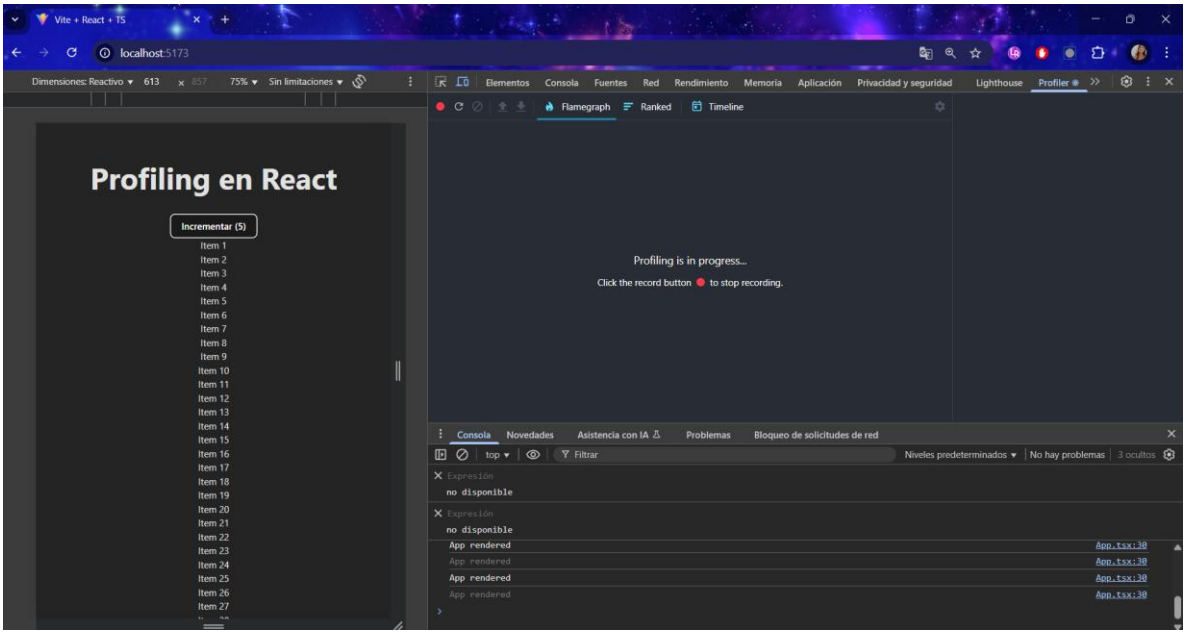
Es importante notar que 138ms es una porción muy grande del tiempo total del commit del App.

Conclusión:

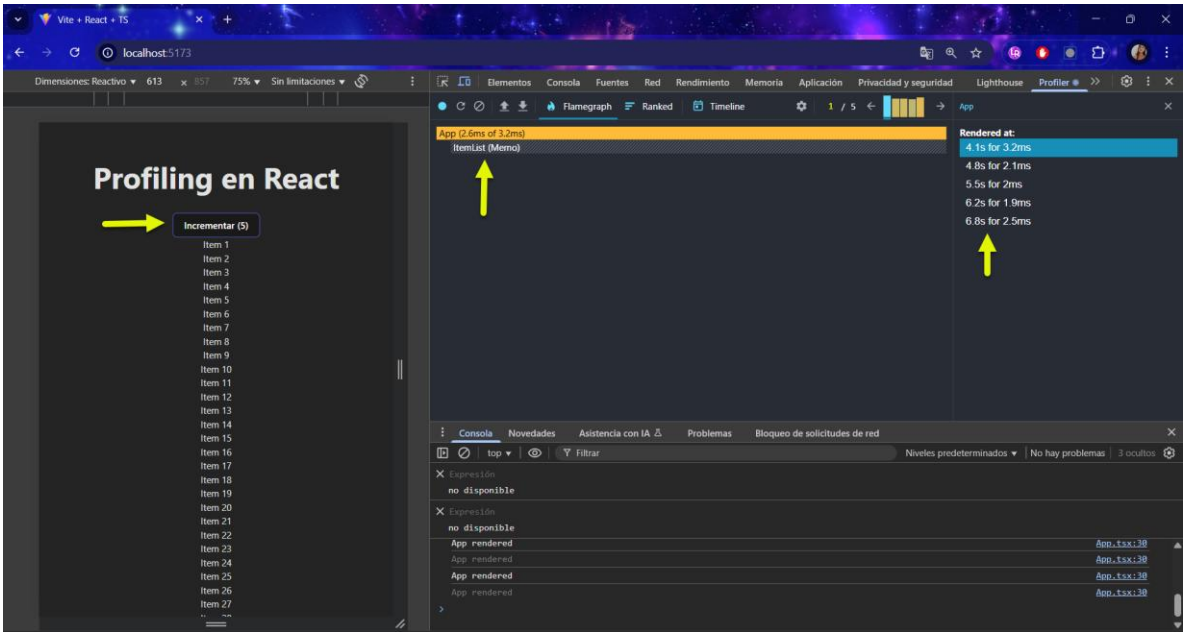
- La captura de pantalla demuestra claramente que el componente **ItemList** se está re-renderizando innecesariamente cada vez que se hace clic en el botón "Incrementar".
- Cuando el estado count en App cambia, App se re-renderiza. Por defecto, React re-renderiza todos los componentes hijos cuando un componente padre se re-renderiza, a menos que se implementen optimizaciones.
- Aunque la prop items de ItemList no ha cambiado, el componente ItemList (y sus 1000 hijos) se vuelven a procesar y renderizar. Esto consume tiempo de CPU (en este caso, unos significativos 138ms) y recursos, lo cual es ineficiente, especialmente si ItemList fuera un componente más complejo o si la lista fuera aún más grande.

Mejoras con Memoización:

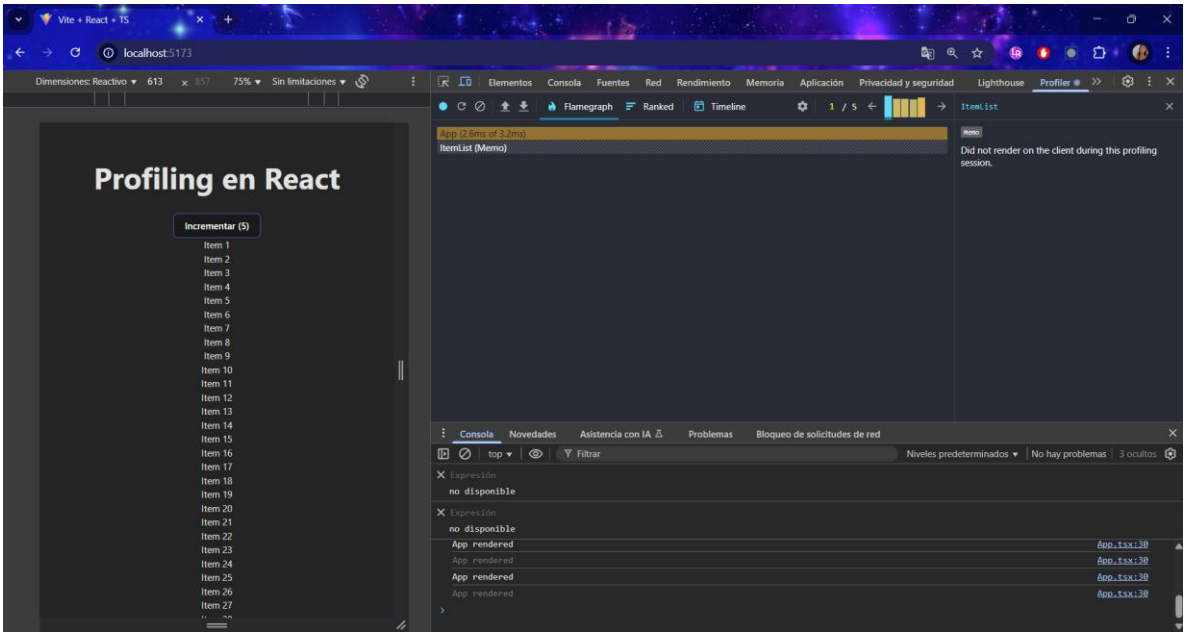
Esto significa que con la memoización, React saltará el renderizado del componente si sus props no han cambiado.



Resultados en la ejecución:



Con ItemList:



Análisis:

Realizamos el análisis de un commit que ocurrió después de hacer clic en "Incrementar". En la segunda imagen, se ve la lista de "Rendered at:", y los tiempos de renderizado para App son muy bajos (ej. 3.2ms, 2.1ms, 2ms), lo cual no demuestra la optimización gracias a la memoización.

Flamegraph:

La barra correspondiente a ItemList (que ahora es MemoizedItemList en el código) aparece en gris.

En la primera imagen, a la derecha del componente ItemList (Memo) en el Flamegraph, se ve una etiqueta "Memo" y el texto: "Did not render on the client during this profiling session.", esto significa que React se saltó el re-renderizado del componente ItemList.

Conclusión:

- Las capturas de pantalla después de aplicar React.memo demuestran exitosamente que la optimización ha funcionado y se ha evitado el re-renderizado innecesario del componente ItemList.

- React.memo compara superficialmente las props de ItemList. Como la prop items no cambia cuando solo se actualiza count (sigue siendo la misma referencia de array), React.memo previene que ItemList se re-renderice.

- Se ha mejorado el rendimiento de la aplicación al eliminar el trabajo innecesario de re-renderizar ItemList y sus 1000 hijos cada vez que el contador se incrementa. Esto libera recursos de CPU y hace que la interfaz de usuario sea más receptiva, especialmente si ItemList fuera un componente más complejo.