Event Sourcing

Integrantes:

Ramón Vila Ferreres: UO272582
Laura Delgado Álvarez: UO271314
Daniel Pérez Prádanos: UO269984

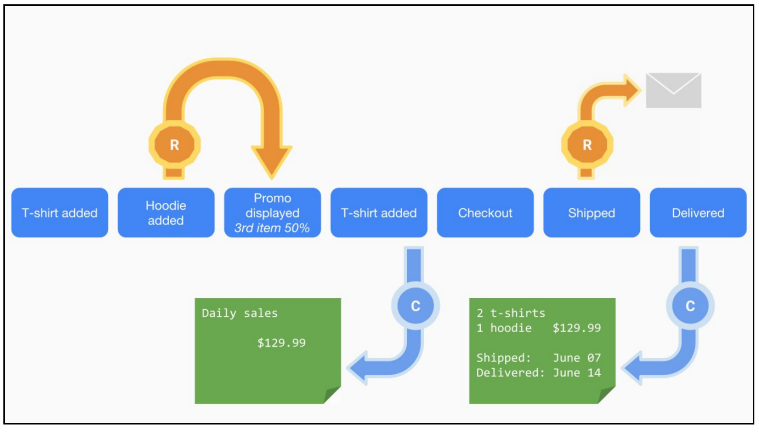
¿Qué es y cómo funciona el *Event Sourcing*?

Event Sourcing puede definirse comparándolo con git, que permite registrar de manera controlada las modificaciones que se realizan sobre cientos de archivos. Así, el uso de este patrón permite registrar los diferentes *eventos* que lanza una aplicación o sistema durante su ejecución para, posteriormente, poder reconstruir su *estado* inicial repitiendo la secuencia de eventos.

Formalmente podría definirse como un patrón que implementa un sistema que registra de manera centralizada todos los cambios en el estado de una aplicación junto con el origen y el momento en el que ocurrieron.

Respecto a su funcionamiento, este proporciona una serie de elementos cuya finalidad principal es simplificar el tratamiento de los datos generados por el flujo de eventos, entre los más importantes se encuentran:

1. **Agregadores.** Representan de manera *resumida (agregada)* el estado actual de la aplicación.
2. **Calculadores.** Leen una secuencia de eventos y los modifican como sea necesario.
3. **Reactores.** Reaccionan a los eventos a medida que ocurren.



A la izquierda se muestra una representación de los cambios en el estado de una aplicación de compra de ropa.

Los elementos azules representan **estados**, mientras que los círculos naranjas representan **reactores** y los azules **calculadores**, cuyos resultados se muestran en las notas verdes.

Aplicaciones potenciales

Existen una serie de situaciones en las que la aplicación de este patrón puede resultar especialmente beneficiosa, algunas son:

1. Cuando una tarea **implica varios pasos** y en caso de error, para revertir los

cambios sería necesario reproducirlos de nuevo en orden para devolver los datos a un estado coherente.
2. Cuando es necesario **capturar la intención**, el **propósito** o el **motivo** en los datos. Por ejemplo, los cambios en una entidad de cliente se pueden capturar como una serie de tipos de evento específicos como Cambio de domicilio, Cierre de cuenta o Fallecimiento.
3. Cuando el uso de **eventos** es una característica natural de la operación de la aplicación y requiere poco esfuerzo de implementación o desarrollo adicional.

¿Qué nos permite el Event Sourcing?

Las acciones de usuario, llamadas a APIs, métodos, etc... generan eventos.
El hecho de capturar un evento, nos permite guardar el estado de la aplicación mediante instantáneas, y un registro de eventos, que nos permitiría saber tanto su estado actual como sus estados previos.

La clave estaría en que podemos garantizar que todos los cambios en los objetos del dominio serán iniciados por los objetos de evento.

Podríamos hacer:
- **Reconstrucción completa**: se basa en descartar el estado completo de la aplicación y reconstruirlo volviendo a ejecutar los eventos del registro de eventos en una aplicación vacía
- **Consulta temporal**: significa que podemos determinar el estado de la aplicación en cualquier momento. Sobre un estado en blanco, ejecutaríamos los eventos hasta cierto momento o evento.
- **Reproducción eventos:** esto se refiere a los eventos incorrectos. Podríamos revertir tanto el evento erróneo como los posteriores y luego reproducirlos de nuevo partiendo del evento corregido.

Ventajas más relevantes:
- El **desacoplamiento** entre el productor del evento y el consumidor del mismo permitiría añadir, tanto productores como consumidores sin problema. Para publicar un evento no sería necesario terminar de procesar otro y una vez publicado podría consumirse en cualquier momento.
- La **trazabilidad** se refiere a la posibilidad de determinar el estado de la aplicación en todo momento.
- La **escalabilidad**, en las bases de datos, sólo podemos añadir elementos, entonces ganamos velocidad. La aplicación enseguida crece horizontalmente.
- La **tolerancia a fallos y reconstrucción**.
- La posibilidad de **realizar instantáneas en cualquier momento en paralelo sin tener que apagar la aplicación en ejecución**.

Sistemas aplicados:
- **Sistemas de control de versiones.**
- **Sistemas de bases de datos.**

Inconvenientes

Una vez que hemos visto que podemos hacer con event sourcing vamos a ver qué inconvenientes tiene.

El principal problema del event sourcing es la complicación extra a la que nos va a forzar en nuestra aplicación. Para usar event sourcing vamos a tener que gestionar la creación de los eventos, su guardado en algún medio de almacenamiento persistente, y el procesamiento de estos, ya sea para modificar el estado de la aplicación o para analizar el "historial" de estados del sistema.

Como estamos viendo incrementa considerablemente la complejidad de la aplicación, y la cantidad de elementos que hay que implementar.

Otro punto en contra del event sourcing es el gran acoplamiento de todo el sistema a los eventos. En caso de que los eventos tengan que cambiar habrá que cambiar todos los módulos que generan los eventos, el sistema que los almacena, y los módulos que los procesan.

En el caso de que tengamos una aplicación que interacciona con un sistema externo, que no conoce nuestro event sourcing, y que de alguna forma esté relacionado con el estado de nuestra aplicación tenemos un problema. Cuando queramos usar los eventos para volver a un estado anterior de la aplicación vamos a tener complicaciones para avisar al sistema externo ya que es posible que no soporte esa vuelta atrás en el estado.

Por lo tanto, aunque el event sourcing tiene ventajas muy considerables también nos complica, a veces en exceso, el sistema.

¿Pasará de moda?

Actualmente hay infinidad de tecnologías muy efímeras, que ganan una gran aceptación rápidamente, pero que al poco tiempo son sustituidas. Esto no tiene por que pasar con el event sourcing, ya que no es dependiente de ningún framework o tecnología concreta, es un patrón arquitectónico de alto nivel. En un futuro se adaptará a las tecnologías venideras, pero manteniendo la filosofía de almacenar el estado de la aplicación como una secuencia de eventos.

Bibliografía:

https://martinfowler.com/eaaDev/EventSourcing.html

https://medium.com/tech-at-nordstrom/adventures-in-event-sourced-architecture-part-1-cc21d06187c7

https://chriskiehl.com/article/event-sourcing-is-hard

https://docs.microsoft.com/es-es/azure/architecture/patterns/event-sourcing

https://kickstarter.engineering/event-sourcing-made-simple-4a2625113224