

Trabajo Fin de Grado

Grado en Ingeniería XXX

Título del Trabajo Fin de Grado

Autor: Nombre Apellidos

Tutor: Nombre Apellidos

Dpto. de Ingeniería XXX
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2026



Trabajo Fin de Grado
Grado en Ingeniería XXX

Título del Trabajo Fin de Grado

Autor:
Nombre Apellidos

Tutor:
Nombre Apellidos
Profesor Titular de Universidad

Dpto. de Ingeniería XXX
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2026

Agradecimientos

Aquí puede agradecer la ayuda recibida durante la realización del trabajo: a su tutor o tutora, a los compañeros, a la familia y a cuantas personas e instituciones hayan contribuido a que el trabajo llegue a buen término.

Resumen

El resumen ofrece al lector una visión clara y concisa de todo el trabajo: el contexto y el problema abordado, el objetivo, el método seguido, los principales resultados y las conclusiones. Debe ser autocontenido y tener una extensión de entre 150 y 500 palabras.

Abstract

The abstract is an English translation of the summary above, including the title of the thesis.

Índice

Agradecimientos	I
Resumen	III
Abstract	V
Introducción	1
1 Edición colaborativa de documentos	3
1.1 Requisitos del sistema	3
2 Enfoques de sincronización	5
2.1 Transformación operacional	5
2.1.1 Función de transformación	5
2.2 Tipos de datos replicados libres de conflictos	5
3 Diseño e implementación	7
3.1 Almacenamiento del historial	7
4 Evaluación	9
4.1 Discusión de los resultados	10
Conclusiones	11
Índice de Figuras	13
Índice de Tablas	15
Índice de Códigos	17
Glosario	19
Bibliografía	21

Introducción

Hoy en día, los documentos de texto rara vez son obra de un único autor. Los estudiantes redactan sus trabajos de curso en parejas, los desarrolladores mantienen conjuntamente la documentación técnica y los equipos de investigación terminan sus artículos en el último momento desde varios lugares a la vez. Para que este tipo de escritura funcione, el editor debe garantizar que todos los participantes vean el mismo texto y que no se pierda ninguna modificación.

El objetivo de este trabajo es comparar los dos enfoques principales para la sincronización de documentos en la edición colaborativa en tiempo real, la transformación operacional y los tipos de datos replicados libres de conflictos, y validarlos en un prototipo de editor web. El primer capítulo delimita los requisitos de un sistema de este tipo, el segundo describe ambos enfoques, el tercero el diseño del prototipo y el cuarto los resultados de las mediciones.

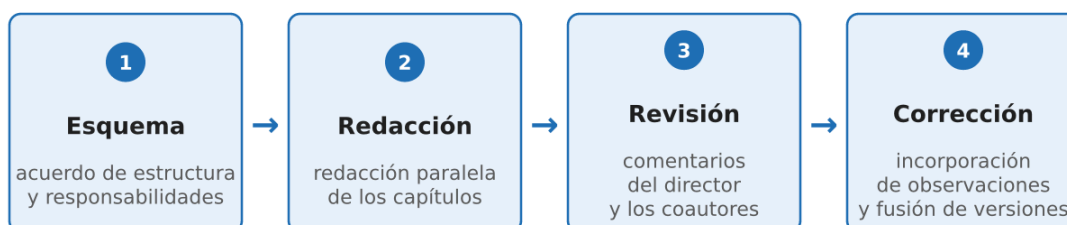
1 Edición colaborativa de documentos

Por edición colaborativa entendemos la situación en la que varios usuarios modifican el mismo documento y sus cambios se propagan mutuamente sin fusión manual. Según la rapidez con que se propagan los cambios, distinguimos tres modos de trabajo básicos:

- **edición asíncrona**, en la que los autores se intercambian versiones completas del documento y solo las fusionan a posteriori;
- **edición con bloqueos**, en la que solo un autor a la vez puede modificar una parte determinada del documento;
- **edición en tiempo real**, en la que cada cambio se refleja de inmediato para los demás usuarios.

Este trabajo se ocupa exclusivamente del último modo. Es el más cómodo para los usuarios, pero el que impone mayores exigencias al diseño del sistema, ya que los cambios surgen simultáneamente en muchos lugares y llegan en distinto orden.

Además, la edición en tiempo real no solo cambia la técnica, sino también el desarrollo de toda la colaboración. El proceso típico de escritura colaborativa, tal como lo describieron los usuarios entrevistados, se recoge en la Figura 1.1.



los pasos 2 a 4 se repiten mientras la revisión aporte observaciones sustanciales

Figura 1.1: Desarrollo típico de la escritura colaborativa de un documento

1.1. Requisitos del sistema

A partir de entrevistas con usuarios de editores existentes y de la bibliografía especializada, hemos derivado cuatro requisitos que debe cumplir un sistema de edición colaborativa:

1. convergencia, es decir, una vez entregados todos los cambios, todas las copias del documento tienen el mismo contenido;

2. preservación de la intención, es decir, el resultado de una modificación corresponde a lo que el autor veía en el momento de realizarla;
3. capacidad de respuesta, es decir, una modificación local se refleja de inmediato sin esperar al servidor;
4. robustez, es decir, una breve interrupción de la conexión no provoca la pérdida del texto a medio escribir.

Los dos primeros requisitos se refieren a la corrección y los dos restantes, a la usabilidad. En la práctica se observa que precisamente la combinación de respuesta inmediata y convergencia garantizada es la parte más difícil de todo el diseño.

2 Enfoques de sincronización

Los sistemas de edición colaborativa en tiempo real aplican primero cada cambio de forma local y solo después lo difunden a los demás participantes. Se diferencian sobre todo en cómo resuelven la situación en la que dos cambios surgen simultáneamente y cada participante los recibe en un orden distinto.

2.1. Transformación operacional

La transformación operacional parte del trabajo de Ellis y Gibbs [1], quienes describieron por primera vez cómo ajustar una operación concurrente para que pueda ejecutarse incluso después de otra operación que su autor desconocía. Cada modificación se registra como una operación de inserción o de borrado en una posición determinada del texto.

Si, por ejemplo, un usuario inserta una palabra al principio de un párrafo y otro borra simultáneamente una palabra al final de este, la posición del borrado debe desplazarse tantos caracteres como tenga la palabra insertada. Este desplazamiento lo calcula la función de transformación.

2.1.1. Función de transformación

La función de transformación recibe un par de operaciones concurrentes y devuelve sus versiones ajustadas. Para que todas las copias del documento converjan, la función debe cumplir las denominadas propiedades de transformación, cuyo cumplimiento es difícil de demostrar para operaciones generales. Por ello, la mayoría de los sistemas desplegados recurren a un servidor central que determina un orden único de las operaciones.

2.2. Tipos de datos replicados libres de conflictos

El segundo enfoque lo constituyen los tipos de datos replicados libres de conflictos, designados en conjunto con la sigla CRDT [2]. En lugar de ajustar las operaciones a posteriori, opta por una representación del documento en la que las operaciones concurrentes conmutan y su orden no influye en el resultado.

Un tipo de datos replicado está libre de conflictos si dos réplicas cualesquiera que han recibido el mismo conjunto de actualizaciones tienen el mismo estado, independientemente del orden en que se les hayan entregado.

Cada carácter del documento recibe un identificador único que permanece invariable durante toda la existencia del carácter. La posición de inserción no se determina entonces mediante un número, sino mediante una referencia al identificador del carácter vecino, de modo que las modificaciones concurrentes en otras partes del texto no pueden desplazarla.

3 Diseño e implementación

El prototipo del editor consta de un cliente web, un servidor de sincronización y una base de datos en la que el servidor almacena de forma continua el estado de los documentos. Tanto el cliente como el servidor están escritos de modo que sea posible sustituir la capa de sincronización y comparar ambos enfoques descritos sobre la misma interfaz de usuario.

El cliente mantiene una réplica local del documento y convierte cada modificación del usuario en una operación sobre el tipo de datos replicado. Las operaciones se envían al servidor por lotes para no saturar la red durante una escritura rápida. El siguiente listado muestra el procesamiento simplificado de un lote entrante en el servidor:

```
def procesar_lote(documento, lote):
    for operacion in lote.operaciones:
        if operacion.id in documento.aplicadas:
            continue
        documento.aplicar(operacion)
        documento.aplicadas.add(operacion.id)
    difundir_a_los_demas(documento.id, lote)
```

Dado que las operaciones conmutan, el servidor no necesita ordenar ni transformar los lotes. Basta con que garantice que ninguna operación se ejecute dos veces y que cada lote llegue a todos los clientes conectados.

3.1. Almacenamiento del historial

El servidor no almacena solo el texto resultante, sino todo el registro de operaciones. A partir del registro es posible restaurar cualquier versión anterior del documento y reconstruir el estado de la réplica de un cliente que ha permanecido desconectado durante un tiempo prolongado. Este enfoque corresponde a una arquitectura basada en un registro inmutable de eventos, tal como la describe Kleppmann [3]. Para que el registro no crezca indefinidamente, el servidor lo compacta a intervalos regulares en instantáneas.

4 Evaluación

Comparamos el prototipo con una implementación de referencia de la transformación operacional sobre un conjunto de registros de sesiones de edición reales. Para cada sesión medimos la latencia, es decir, el tiempo transcurrido entre una modificación en el equipo del autor y su visualización en los de los demás participantes, y calculamos la latencia media de la sesión según (4.1):

$$\bar{t} = \frac{1}{n} \sum_{i=1}^n (t_i^{\text{entrega}} - t_i^{\text{modificación}}) \quad (4.1)$$

donde n es el número de operaciones de la sesión. Los valores medidos para distintos números de usuarios trabajando simultáneamente se resumen en la Tabla 4.1.

Tabla 4.1: Latencia media y tamaño de los datos almacenados según el número de usuarios

Usuarios	Latencia OT (ms)	Latencia CRDT (ms)	Datos (kB)
2	38	35	12
5	52	41	19
10	87	48	31
20	164	63	58

A medida que aumenta el número de usuarios, la latencia de la transformación operacional crece mucho más deprisa, ya que el servidor debe transformar cada operación frente a todas las operaciones concurrentes. El precio de la menor latencia de los CRDT es un mayor volumen de datos almacenados, puesto que cada carácter lleva su identificador. La distribución temporal de las modificaciones, en la que se basa la prueba de carga, se muestra en la Figura 4.1.

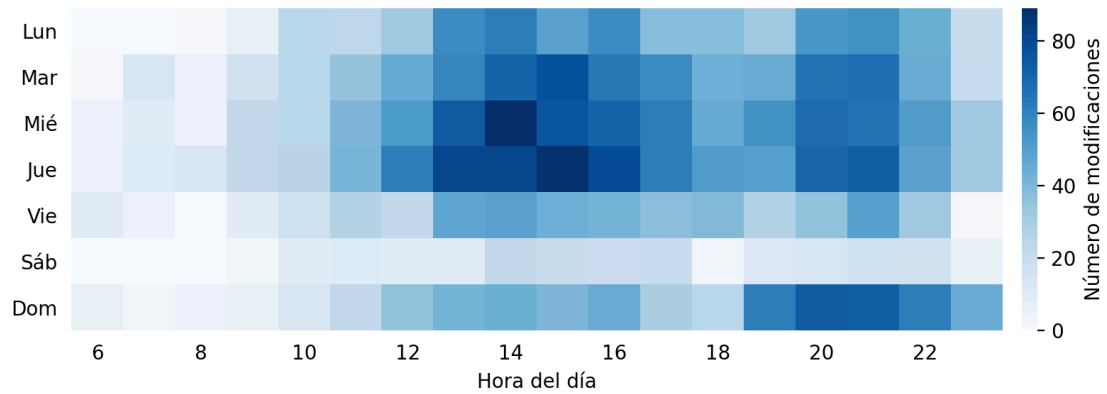


Figura 4.1: Número de modificaciones por día de la semana y hora en las sesiones analizadas

4.1. Discusión de los resultados

Los resultados confirman que, para equipos pequeños, ambos enfoques son equivalentes y que lo decisivo es más bien la sencillez de la implementación. Con un mayor número de usuarios simultáneos, los CRDT presentan una clara ventaja tanto en latencia como en robustez frente a las interrupciones de la conexión. Las entrevistas complementarias con usuarios, realizadas conforme a los principios de la investigación cualitativa [4], mostraron que los usuarios no perciben una diferencia de latencia inferior a cien milisegundos, pero consideran un problema grave la pérdida del texto a medio escribir tras una interrupción de la conexión.

Conclusiones

Este trabajo ha comparado la transformación operacional y los tipos de datos replicados libres de conflictos como dos enfoques para la sincronización de documentos en la edición colaborativa en tiempo real. Mediante un prototipo de editor web hemos mostrado que los CRDT permiten un servidor más sencillo, sin ordenación central de las operaciones, y ofrecen una latencia menor cuando el número de usuarios es mayor.

La principal desventaja de los CRDT sigue siendo el mayor volumen de datos almacenados. Por ello, el siguiente paso será comprobar cómo influye la compactación periódica del registro en instantáneas en el tamaño de los datos y cómo se comporta el prototipo en un funcionamiento prolongado con decenas de miles de documentos.

Índice de Figuras

Figura 1.1	Desarrollo típico de la escritura colaborativa de un documento	3
Figura 4.1	Número de modificaciones por día de la semana y hora en las sesiones analizadas	10

Índice de Tablas

Tabla 4.1 Latencia media y tamaño de los datos almacenados según el número de usuarios

9

Índice de Códigos

Glosario

TCP Transmission Control Protocol

UDP User Datagram Protocol

Bibliografía

- [1] C. A. Ellis y S. J. Gibbs, «Concurrency control in groupware systems», en *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, New York: ACM, 1989, pp. 399-407. doi: 10.1145/67544.66963.
- [2] M. Shapiro, N. Preguiça, C. Baquero, y M. Zawirski, «Conflict-free replicated data types», en *Stabilization, Safety, and Security of Distributed Systems*, Berlin: Springer, 2011, pp. 386-400. doi: 10.1007/978-3-642-24550-3_29.
- [3] M. Kleppmann, *Designing data-intensive applications*. Sebastopol: O'Reilly Media, 2017.
- [4] J. Hendl, *Kvalitativní výzkum: základní metody a aplikace*. Praha: Portál, 2005.