Desafíos en el Diseño y la Comunicación entre Microfrontend

Franco Pappatella, Natalia Cirillo, Nicolas Battaglia, Gustavo Rossi

CAETI. Facultad de Tecnología Informática, Universidad Abierta Interamericana (UAI)
pappatella.francoariel@gmail.com, nataliacirilloo@gmail.com, nicolas.battaglia@uai.edu.ar, gustavo.rossi@uai.edu.ar

RESUMEN

Los Microfrontend se consolidaron como un enfoque arquitectónico para aplicaciones interactivas distribuidas al dividir la interfaz en componentes independientes desarrollados por distintos equipos y tecnologías. Esta autonomía mejora la escalabilidad, pero introduce el desafío de la comunicación entre módulos.

Este proyecto busca analizar y clasificar las estrategias de comunicación existentes entre Microfrontend y sus implicancias en desacoplamiento, autonomía y complejidad, aportando criterios para su selección en arquitecturas interactivas distribuidas.

Palabras clave: *Microfrontends, arquitecturas distribuidas, comunicación entre componentes, desacoplamiento, front-end modular*

CONTEXTO

Esta investigación representa una línea temática dentro del proyecto: Aspectos de Arquitectura y diseño en aplicaciones distribuidas (Web, Móviles, IoT, etc) con foco en Microservicios (Battaglia et al 2025ª)

La investigación se enmarca en el área de Arquitecturas de Software y Sistemas Distribuidos, enfocada en patrones aplicados al front-end de aplicaciones distribuidas complejas.

La necesidad de escalabilidad y despliegue independiente impulsó la adopción de Microfrontend. Este tipo de descomposición plantea desafíos en la gestión del estado compartido y la coordinación entre módulos, para mantener la independencia sin afectar la coherencia del sistema. Asimismo los Microfrontend plantean otros desafíos de diseño interesantes respecto al diseño de la interfaz y la interacción

1. INTRODUCCIÓN

Las arquitecturas de Microservicios representaron un avance importante en cuanto a la obtención de modularidad en aplicaciones complejas. Descomponer un dominio de aplicación en componentes independientes (los microservicios) que se comunican entre sí permite (cuando los componentes han sido diseñados apropiadamente) maximizar la modularidad, garantizando mayor eficiencia no solamente en el proceso de evolución del software sino también en su despliegue (Jamshidi et al. 2018, Kapferer et al. 2020, Newman 2021, Oumoussa et al. 2024).

Hasta hace unos años este tipo de descomposición se aplicaba fundamentalmente a los aspectos del dominio de aplicación (por ejemplo descomponiendo en función de “áreas” del negocio), pero no a los aspectos de presentación e interacción.

Los Microfrontend también proponen descomponer un front-end monolítico en

partes desarrollables y desplegables de forma independiente, habilitando autonomía de equipos y flexibilidad tecnológica, aunque con costos como mayor complejidad operativa, desafíos de gestión de estado y duplicación de código (Silva et al., 2025).

Los Microfrontend aplican los mismos conceptos de descomposición en componentes, pero en el dominio de la interfaz y la interacción, dividiendo la interfaz en módulos autónomos (por ejemplo, por dominio funcional), que, de forma semejante a los microservicios, podrán ser desarrollados y desplegados de forma independiente.

Si bien es esperable obtener las mismas ventajas que se han obtenido en el uso de microservicios, existen algunos problemas que aun deben ser caracterizados y resueltos. Por un lado la descomposición de interfases en módulos independientes durante el proceso de diseño no necesariamente resulta trivial dado que los diseñadores (de interfaz) suelen ver a la misma como un desarrollo holístico, que debe resultar en interfaces consistentes. Con este análisis, entonces, la división en módulos propuesta por los Microfrontend (eventualmente dividiendo también los aspectos estéticos en equipos de desarrollo diferentes), podría no ser la mejor.

Por otro lado, la independencia de estas componentes requiere el establecimiento de mecanismos de comunicación, que exceden el tipo de mecanismos utilizados por los microservicios, dada la propia naturaleza de los Microfrontend. Mientras que en arquitecturas monolíticas tradicionales, la relación entre la interfaz y el back-end suele basarse en dependencias directas y estados compartidos (por ejemplo siguiendo el estilo del patrón Model-View-Controller), en entornos distribuidos pueden generar acoplamiento y reproducir limitaciones típicas de los monolitos.

Este problema de acoplamiento no es sólo técnico, sino también organizacional. La autonomía de los Microfrontend no se limita al despliegue independiente, sino que también comprende la capacidad de ser desarrollados y mantenidos por equipos distintos, con control sobre sus decisiones técnicas.

El desacoplamiento del front-end favorece equipos autónomos organizados por rebanadas verticales de funcionalidad, con responsabilidad end-to-end sobre una parte del producto (Jackson, 2019)

Para que esta autonomía sea efectiva, la comunicación entre componentes no puede basarse en el conocimiento de las implementaciones internas de otros Microfrontends. Cuando esto ocurre, se introducen dependencias que obligan a una coordinación constante y reducen la capacidad de evolución independiente.

Por otra parte, no se debe confundir el concepto de Microfrontend con que cada módulo debe obtener todos sus datos de forma independiente, ya que podríamos tener datos duplicados entre los Microfrontend, y eso podría conllevar un derroche innecesario de memoria del navegador. Para evitar esto, es necesario diseñar un flujo de datos eficiente para garantizar una comunicación performante y coherente, formando una estructura de datos capaz de almacenar el estado que tienen en común los Microfrontend de nuestra aplicación.

Ante estos desafíos, la elección de una buena estrategia de comunicación se convierte en un factor clave para implementar esta arquitectura. Evitar el acoplamiento innecesario, maximizar la autonomía de los

módulos y garantizar la eficiencia del flujo de datos requiere emplear mecanismos diseñados específicamente para entornos distribuidos y heterogéneos.

2. LÍNEAS DE INVESTIGACIÓN Y DESARROLLO

En esta línea de investigación buscamos profundizar diversos aspectos aún abiertos en la tecnología de Microfrontend.

Por un lado, nos interesa discutir los problemas de diseño mencionados en la sección anterior, específicamente cómo hacer compatible la naturaleza distribuida del diseño, desarrollo y despliegue de los Microfrontend con el diseño en general centralizado de los estilos de interfaz e interacción en las aplicaciones modernas. Observamos en este punto que existe una tensión en este aspecto entre la modularidad, típica de las aplicaciones distribuidas desarrolladas con microservicios, y la usabilidad deseable en cualquier aplicación moderna y que, según el estado del arte en diseño, suele alcanzarse con interfaces consistentes y desarrolladas como un "todo".

Esta tensión puede derivar en anti-patrones de descomposición: nano frontends (granularidad excesiva y mayor costo/acoplamiento) o mega frontends (granularidad insuficiente y "efecto monolito") (Silva et al., 2025).

En los aspectos de comunicación investigaremos sobre las diferentes estrategias de comunicación entre Microfrontend y su impacto en el desacoplamiento y la evolución independiente dado que la elección del mecanismo de comunicación debe responder a criterios técnicos y organizacionales, priorizando la evolución independiente y la coherencia arquitectónica.

3. RESULTADOS OBTENIDOS / ESPERADOS

En un análisis inicial hemos observado lo siguiente respecto a algunos de los mecanismos de comunicación más comunes, expresados por los patrones: Event Bus, Local Storage, Backend for Front-end y Callbacks.

El patrón **Event Bus** centraliza la comunicación mediante eventos compartidos. Los módulos emiten y escuchan sin dependencia directa, favoreciendo el desacoplamiento. Sin embargo, los eventos no persisten y requieren convenciones.

El uso de **Local Storage** permite persistencia y sincronización entre pestañas, útil para información global, aunque con limitaciones en reactividad.

El patrón **Backend For Front-end** (BFF) introduce una capa intermedia entre Microfrontend y backend. Cada módulo interactúa con un backend propio, reduciendo la complejidad en el cliente y favoreciendo la autonomía, aunque incrementa la complejidad general.

Los **callbacks** orquestados por un contenedor centralizan el estado y el flujo de información. Es un enfoque simple, pero limita la independencia tecnológica.

En estos momentos nos encontramos analizando un repertorio más completo de estilos existentes y las características de cada uno.

En la tradición de los patrones de diseño y arquitectural esperamos desarrollar una guía exhaustiva con recomendaciones de cuándo usar cada tipo de mecanismo y casos de éxito, junto con discusiones respecto a trade-offs en cada caso.

Otro resultado esperado es un conjunto de guidelines de diseño para buscar la compatibilidad entre desacoplamiento del diseño y usabilidad de la interfaz.

4. FORMACIÓN DE RECURSOS HUMANOS

Este trabajo está originado en la participación de estudiantes de grado de la Facultad de Tecnología Informática de la UAI en el proyecto de investigación Aspectos de Arquitectura y diseño en aplicaciones distribuidas (Web, Móviles, IoT, etc) con foco en Microservicios. Se espera que parte de esta investigación termine en trabajos de fin de carrera. Al mismo tiempo se están extendiendo dos tesis doctorales, sobre diseño de microservicios usando métodos ágiles (Battaglia et al 2024, Battaglia 2025, Battaglia et al 2025b) e inteligencia artificial generativa (Narvaez 2025) de manera de definir proyectos finales de maestría y tesis doctorales alrededor de esta nueva temática.

5. BIBLIOGRAFÍA

Battaglia, N., Garcia, A. N., & Congiusti, A. (2024). Descubrimiento de Microservicios en Metodologías Ágiles: un mapeo sistemático de la literatura. XXX Congreso Argentino de Ciencias de La Computación, 1506–1510.

Battaglia, N. (2025). Identificación Ágil de Microservicios Utilizando DDD y BDD. XXVIII Ibero-American Conference on Software Engineering (CIbSE 2025 Doctoral Symposium).

Battaglia, N., Narváez, D., & Rossi, G. H. (2025a). Aspectos de arquitectura y diseño en aplicaciones distribuidas (web, móviles, Iot, etc) con foco en microservicios. In XXVII Workshop de Investigadores en Ciencias de la Computación (Mendoza, 10 y 11 de abril de 2025)

Battaglia, N., Rossi, G., & Fernández., A. (2025b). Behavior-Driven Microservice Architecture: Un Marco Metodológico para la Identificación Iterativa de Microservicios en Proyectos Ágiles Greenfield. Jornadas Iberoamericanas de Ingeniería de Software e Ingeniería Del Conocimiento JIISIC 2025.

Fernandez, T. (2022, 29 de Septiembre). *Microfrontend: Microservices for the Front-end*. Semaphore. https://semaphore.io/blog/Microfrontend

Jackson, C. (2019, 19 de Junio). *Micro front-ends*. Martin Fowler. https://martinfowler.com/articles/micro-front-ends.html

Jamshidi, P., Pahl, C., Mendonça Nane Kratzke, & others. (2018). Microservices: The Journey So Far and Challenges Ahead. IEEE Software, 35(3), 24–35.

Kapferer, S., & Zimmermann, O. (2020). Domain-specific Language and Tools for Strategic Domain-driven Design, Context Mapping and Bounded Context Modeling. MODELSWARD, 299–306.

Narvaez D. Explorando el Uso de Inteligencia Artificial en el Descubrimiento y Diseño de Microservicios. In: Congresso Ibero-Americano em Engenharia de Software (CIbSE). SBC. 2025:224–31.

Newman, S. (2021). Building Microservices (2nd ed.). O'Reilly Media.

Oumoussa, I., & Saidi, R. (2024). Evolution of microservices identification in monolith

decomposition: A systematic review. IEEE Access, 12, 23389–23405.

Silva, N., Rodrigues, E., & Conte, T. (2025, 27 de Marzo). *A Catalog of Micro Front-ends Anti-patterns*. Federal University of Amazonas. https://arxiv.org/abs/2411.19472