Micro-Frontends: tendencia en el desarrollo de aplicaciones web

Milton Vindas Rodríguez
mvindasr@ucenfotec.ac.cr

Universidad Cenfotec
Bachillerato en Ingeniería del Software

Abstract
Nuevos frameworks de frontend se introducen continuamente en el mercado y los desarrolladores tienen muchas opciones válidas para crear potentes aplicaciones web ricas en funciones, como las aplicaciones de una sola página (SPA), aplicaciones que se crean del lado del servidor (SSR) o archivos HTML estáticos combinados en una página web. Sin embargo, la mayoría de estos terminan siendo frontends monolíticos. Por lo tanto, el lado cliente de la aplicación crece y su desarrollo se vuelve difícil de escalar, especialmente si diferentes equipos necesitan editar la misma aplicación de frontend simultáneamente. Los micro-frontend se introdujeron para permitir la descomposición del frontend en pequeños frontends individuales y semi-independientes, separando la lógica empresarial del frontend y creando servicios independientes que interactúan juntos.

Palabras Clave
Micro-frontends, aplicación, página web, monolito, framework, arquitectura

Introducción

Las aplicaciones web, como sistemas de software, se dividen tradicionalmente en dos partes: backend (servidor), que en muchos casos es responsable del procesamiento de datos, y frontend (cliente), que se necesita para proporcionar una interfaz conveniente para la interacción entre los usuarios y el sistema. Los componentes de backend podrían desarrollarse utilizando diferentes enfoques arquitectónicos, uno de ellos es el de Microservicios, uno de los más adecuados para sistemas escalables, que se enfoca en el concepto de separación de partes lógicamente independientes de un sistema. Se utiliza en lugar de estilos arquitectónicos monolíticos, porque estos últimos son difíciles de escalar y presentan dificultades para desarrollar diferentes partes de él simultáneamente. En el mundo de las aplicaciones del lado del cliente, su tamaño, así como su complejidad, han venido creciendo en los últimos años, especialmente en el desarrollo web. Debido a esto, la arquitectura empieza a jugar un papel más importante. La comunidad mundial de frontend introdujo la idea de micro-frontends, que es similar al concepto de microservicios e implica la separación de una gran aplicación en partes más pequeñas que conduce a la posibilidad de desarrollo concurrente, un tiempo de descarga más rápido y un mayor rendimiento. El objetivo principal de esta revisión es examinar las generalidades, ventajas y desventajas de los micro-frontends en la implementación de aplicaciones web.

Desarrollo del tema

La estrategia monolítica del software ha sido desafiada en los últimos años. En un sistema, la base del código puede ser tan grande que no se puede mantener. Los primeros avances para afrontar este

problema fueron la modularización y las soluciones basadas en componentes, que condujeron al paradigma de la arquitectura orientada a servicios. Recientemente, ha evolucionado aún más a una arquitectura de microservicio, un enfoque intuitivo, con la que se dividió la lógica del backend en servicios independientes de menor tamaño. Cada servicio es accesible a través de la API, utilizando el principio REST. Esta arquitectura se volvió cada vez más popular, sin embargo, una descomposición similar no se aplicó al frontend (Figura 1).

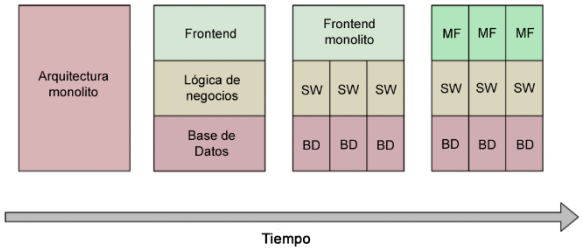

Figura 1. Evolución de las arquitecturas web desde el punto de vista del frontend: Monolito - 3 capas - Microservicios / SOA – Micro-frontends. Abreviaturas: WS (servicio web), DB (base de datos).

Hoy en día, las aplicaciones de una sola página (SPA) que se ejecutan sobre una arquitectura de microservicio son las más utilizadas junto con las aplicaciones ensambladas del lado del servidor (SSR) y las páginas web formadas mediante la combinación de páginas HTML estáticas. Aunque estos marcos son opciones aceptables para crear aplicaciones web potentes y ricas en funciones, tienden a ser interfaces monolíticas, lo que aumenta el tamaño de la aplicación del lado del cliente y limita la escalabilidad del desarrollo de la aplicación por parte de varios equipos. Para afrontar este inconveniente, surgió el término de micro-frontend, acuñado por ThoughtWorks Technology Radar (2016). El concepto es una extensión de los microservicios utilizados en backend. Los micro-frontends superan el problema al descomponer el frontend en diferentes características propiedad de equipos independientes (Figura 1), de modo que cada equipo tiene un área de negocios separada en la que el equipo es competente. Varias industrias como DAZN, Ikea, New Relic, SAP y otras han implementado esta arquitectura (Geers, 2020).

Es importante constatar que el concepto no es solo un enfoque arquitectónico para la construcción de aplicaciones web, sino también un enfoque organizacional. En micro-frontends, la capa de presentación se divide en piezas pequeñas y manejables, cada una de las cuales proporciona una característica específica en la aplicación web (Geers, 2020), por ejemplo, un menú o un reproductor de video. Una pieza de frontend solo se comunica con los servicios de backend que necesita para la función que maneja, lo que significa que un sistema de micro frontends se divide en una serie de microaplicaciones de pila completa (Figura 2). Cada microsistema es una aplicación autónoma que tiene su propio *pipeline* de entrega continua que construye, prueba y despliega la micro aplicación, incluso se puede desplegar en línea utilizando diferentes servicios de *deployment* (Jackson, 2019). Como resultado, una microaplicación puede funcionar por sí sola, incluso si otras microaplicaciones no funcionan (Montelius, 2021).

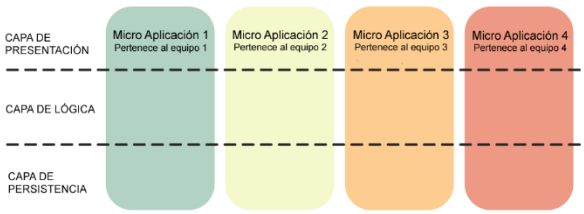

Figura 2. Descripción general de la arquitectura de una aplicación micro-frontends, donde cada color representa una microaplicación de pila completa propiedad de un equipo dedicado. Modificado de Montelius (2021)

Aspectos técnicos de la composición de micro-frontends

Para definir la arquitectura de micro-frontend, la decisión clave a tomar es identificar cómo considerar el micro-frontend desde el punto de vista técnico. Para ello existen dos opciones (Kroiß, 2021; Wang *et al*., 2020):

(i) División horizontal: "múltiples micro-frontends por página". En una división horizontal, se cargan varias aplicaciones más pequeñas en la misma página y esto requiere que varios equipos coordinen sus esfuerzos, ya que cada equipo es responsable de una parte de la vista. Un paso importante en esta opción es definir cómo se comunican entre sí los micro-frontend; un método es utilizar un emisor de eventos inyectado en cada micro-frontend. Esto haría que cada micro-frontend desconociera por completo a sus compañeros y permitiría implementarlos de forma independiente. Cuando un micro-frontend emite un evento, los otros micro-frontends suscritos a ese evento específico reaccionan apropiadamente.

(ii) División vertical: "un micro-frontend por vez". En este escenario, cada equipo es responsable de un dominio empresarial, por ejemplo, la autenticación o pago. Con la división vertical, se aplica un diseño controlado por dominio (DDD). Acá es importante comprender cómo compartir información entre micro-frontends, se debe pensar en cómo se comunican las vistas cuando cambian. Es posible que las variables se pasen a través de una cadena de consulta, o usando la URL para pasar una pequeña cantidad de datos. Alternativamente, es posible utilizar el almacenamiento web para almacenar temporalmente (almacenamiento de sesiones) o permanentemente (almacenamiento local) la información que se compartirá con otras micro-frontends.

Comparación de implementaciones

Por ser un concepto relativamente reciente, la arquitectura de micro-frontends es muy flexible, con múltiples implementaciones. Los esquemas típicos son los siguientes (Wang *et al*., 2020, Yang *et al*., 2019):

(i) Distribución de rutas, que reparte diferentes servicios a diferentes aplicaciones frontend mediante el enrutamiento. Por lo general, se implementa a través del proxy inverso del servidor HTTP o por el enrutamiento proporcionado por el *framework* de la aplicación. Este es el método de implementación más simple, pero como los datos en los diferentes módulos son diferentes, puede llevar más tiempo cargarlos.

ii) Incrustación por <Iframe>, una tecnología antigua y ordinaria, en la cual el sistema integra cada sub-aplicación en sus propios *iframes*, lo que permite que cada aplicación utilice cualquier *framework* que necesite, y las aplicaciones pueden ejecutarse de forma

independiente sin necesidad de herramientas de coordinación y dependencias con otras aplicaciones. La mayor ventaja de usar *iframes* es que permite aislar el entorno de ejecución de componentes y aplicaciones. Por lo tanto, el desarrollo del sistema puede tomar utilizar, por ejemplo, una sección en React, otra en Angular y luego usar JavaScript nativo o cualquier otra tecnología para desarrollar otras partes. Siempre que cada *iframe* provenga de la misma fuente, la mensajería entre ellos es bastante simple y poderosa. La desventaja es los paquetes serán muy grandes y es difícil brindar soporte a varios *iframes* anidados.

(iii) Componentes web son un conjunto de diferentes herramientas, que incluyen elementos personalizados, *shadow DOM* y plantillas e importación HTML (Wusteman, 2019). Los componentes web son reutilizables y con capacidades de encapsulación, por lo que se pueden importar a aplicaciones web de una manera muy elegante. Sin embargo, vuelven a la arquitectura más complicada, ya que la comunicación entre los componentes podría volverse un problema particularmente grande.

Enfoques de composición de micro-frontends

Los tres principales enfoques para ensamblar aplicaciones basadas en micro-frontends (Figura 3) se detallan a continuación (Peltonen *et al*., 2021).

(i) C*omposición del lado del cliente*: un *shell* de aplicación carga los micro-frontends dentro de sí mismo. Los micro-frontends deben tener como punto de entrada un archivo JavaScript o HTML para que el *shell* de la aplicación pueda agregar dinámicamente los nodos DOM en el caso de un archivo HTML o inicializar la aplicación JavaScript cuando el punto de entrada es un archivo JavaScript. Los *frameworks* de frontend React, AngularJS y Vue se han vuelto populares ya que brindan enrutamiento dinámico y una mejor interfaz de usuario. En estos, los micro-frontends se envuelven como componentes web para luego cargarlos en el navegador.

(ii) C*omposición del lado del servidor*: el servidor backend compondrá la vista obteniendo todas los micro-frontends y ensamblando la página web final. Esto ayuda a lograr buenas velocidades de carga que no son posibles con la composición del lado del cliente. Esta técnica es útil cuando se crean micro-frontends que se basan en principios de mejora progresiva.

(iii) C*omposición del lado del borde*: La página web se ensambla en el nivel de una red de entrega de contenidos (CDN). Muchos proveedores de CDN dan la opción de usar un lenguaje de marcado basado en XML llamado Edge Side Include (ESI).

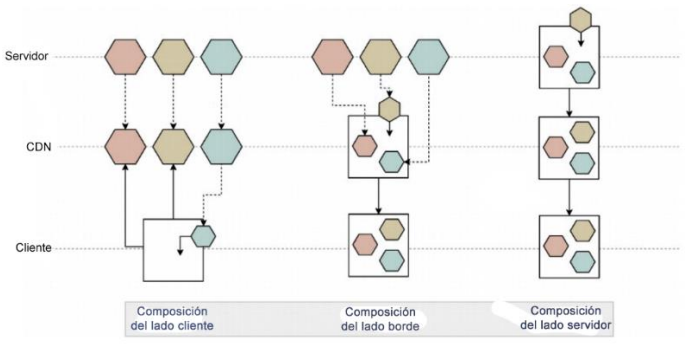

Figura 3. Diferentes enfoques para combinar micro-frontends.

Beneficios de micro-frontends

(i) Independencia del equipo: tienen como objetivo aislar a los equipos y son herramientas eficientes para ayudar a los equipos a trabajar de forma independiente. Cada equipo tiene total autonomía de una porción vertical de la aplicación y puede enfocarse en especializarse en ese dominio.

(ii) Organización de código: la partición efectiva del proyecto en micro-frontends también ayuda a una mejor organización del código del proyecto. Cada micro-frontend se enfoca en una parte más pequeña de la aplicación.

(iii) Genera independencia: permite un desarrollo de funciones más rápido y las partes del proyecto se pueden publicar de forma independiente.

(iv) Pruebas de superficie reducidas y construcciones más rápidas.

(v) Aislamiento de fallas: una de las ventajas importantes de los micro-frontends sobre los monolitos es que, en caso de error, no es necesario rechazar toda la aplicación. Es posible detectar en qué módulo se ha producido el error y se puede realizar una solución adecuada a ese módulo específico.

(vi) Tecnología agnóstica: no depende de los *frameworks* subyacentes utilizados para desarrollar las micro-frontends. Esto agrega el beneficio de incorporar distintos *stacks* de tecnología para diferentes micro-frontends según el conjunto de habilidades existentes en el equipo.

Dificultades de micro-frontends

Como es una arquitectura emergente con diversas implementaciones, existen una serie de limitaciones (Prajwal *et al*., 2021) que los investigadores están intentando remediar proponiendo metodologías innovadoras.

(i) Redundancia: como las micro-frontends abarcan a varios equipos independientes que desarrollan sus pilas en paralelo, podría introducir mucha redundancia en el código JavaScript y CSS.

(ii) Comunicación: Está en contra de los principios de las micro-frontends comunicarse entre dos micro-frontends. Esto puede resultar problemático para las funciones existentes en el sistema.

(iii) Riesgo de divergencia: el código puede comenzar a divergir en gran medida al dividir el proyecto en diferentes micro-frontends y terminar con proyectos completamente diferentes.

(iv) Problemas de desempeño: la velocidad de carga de la primera página es significativamente mayor en comparación con otros enfoques. Esta latencia puede afectar la interfaz de usuario. Sin embargo, si los recursos se almacenan en caché, el rendimiento es mejor que otros enfoques.

¿Cúando utilizar la arquitectura de micro-frontends?

Las características de escalabilidad del equipo y el enfoque estratégico/táctico detalladas anteriormente, indican que los micro-frontends tienen como objetivo facilitar el escalamiento de proyectos. Por lo tanto, un proyecto sin la necesidad de escalarse a sí mismo y a su equipo de desarrollo no se beneficiaría de esto: si un equipo ya es pequeño (y también planea mantenerse pequeño), ya tiene los beneficios de contar con una comunicación y decisiones rápidas. Esta circunstancia lleva a la conclusión de que los micro-frontends tienen sentido para

equipos de tamaño mediano a grande Klimm, 2021). Geers (2020) ejemplifica que un buen tamaño de equipo consistiría en alrededor de 5 personas, basado en la regla de las dos pizzas de Jeff Bezos que establece que “un equipo es demasiado grande cuando dos pizzas grandes no son suficientes para satisfacer al equipo”. Otra característica de los micro-frontends es que esta arquitectura es más adecuada para aplicaciones web, aunque no se limita en teoría a ese contexto. Geers (2020) también explica esto al argumentar que las aplicaciones web son mucho más capaces de implementar arquitecturas complejas en comparación con aplicaciones nativas en *app stores*, ya que estas últimas deben ofrecerse como sistemas cliente-monolíticos que puedan ser revisadas por los propietarios de las tiendas virtuales.

Conclusión

En los últimos años, los micro-frontends han recibido una gran atención por parte del mundo académico y se han convertido en uno de los objetos de investigación clave en el campo de las ciencias de la información. Este estilo arquitectónico es una respuesta de la comunidad a la creciente complejidad de las aplicaciones web. Los micro-frontend están dividiendo cosas grandes y atemorizantes en partes más pequeñas y manejables al extender el concepto de microservicios al frontend. En la actualidad, los micro-frontends aún se encuentran en etapa de exploración y no están lo suficientemente maduros. A pesar de que existen algunos inconvenientes, los micro-frontends tienen beneficios significativos y, por lo tanto, tienen un gran potencial para el desarrollo de la capa de presentación de la aplicación web de un equipo de tamaño mediano-grande.

Referencias

Geers, M. (2020), Micro Frontends in Action, Manning Publications Co., Shelter Island, NY 11964. ISBN 9781617296871 - ePub.

Jackson, C. (2019). Micro Frontends. Recuperado de: https://martinfowler.com/articles/micro-frontends.html.

Klimm, M. C. (2021). Design Systems for Micro Frontends-An Investigation into the Development of Framework-Agnostic Design Systems using Svelte and Tailwind CSS. Recuperado de: https://epb.bibl.th-koeln.de/frontdoor/deliver/index/docId/1666/file/SCIBachelor_Design_Systems_and_Micro_Frontends.pdf

Kroiß, M. (2021). From Backend to Frontend - Case study on adopting Micro Frontends from a Single Page ERP Application monolith. Recuperado de: https://repositum.tuwien.at/bitstream/20.500.12708/17595/1/Kroiss%20Manuel%20-%202021%20-%20From%20Backend%20to%20Frontend%20-%20Case%20study%20on%20adopting%20Micro...pdf

Montelius, A. (2021). An Exploratory Study of Micro Frontends. Recuperado de: https://www.diva-portal.org/smash/get/diva2:1570726/FULLTEXT01.pdf

Peltonen, S., Mezzalira, L., & Taibi, D. (2021). Motivations, benefits, and issues for adopting Micro-Frontends: A Multivocal Literature Review. Information and Software Technology, 136, 106571.

Prajwal, Y.R., Parekh, J.V., & Shettar, D.R. (2021). A Brief Review of Micro-frontends. United International Journal for Research & Technology (UIJRT).

ThoughtWorks. (2016). Micro frontends. Recuperado de: https://www.thoughtworks.com/radar/techniques/micro-frontends.

Wang, D., Yang, D., Zhou, H., Wang, Y., Hong, D., Dong, Q., & Song, S. (2020). A Novel Application of Educational Management

Information System based on Micro Frontends. Procedia Computer Science, 176, 1567-1576.
Wusteman, J. (2019). The potential of web components for libraries. Library Hi Tech. 37(4): 713-720.
Yang, C., Liu, C., & Su, Z. (2019). Research and Application of Micro Frontends. IOP Conference Series: Materials Science and Engineering, 490, 062082.

"Milton Vindas es estudiante de la Universidad Cenfotec y cursa la carrera de Ingeniería del Software. Cuenta con una Ingeniería en Biotecnología del Instituto Tecnológico de Costa Rica y con una maestría en Microbiología de la Universidad de Costa Rica. Actualmente realiza prácticas de aplicaciones de página simple (SPA) utilizando principalmente el framework React"