Micro-Frontends

Diego Villa García UO277188

Juan Mera Menéndez UO277406

Héctor Martín Gutiérrez UO239198

● Explicación general y origen.

El concepto de micro-frontend apareció por primera vez en el informe *Technology Radar* en 2016. El Technology Radar es un informe que realiza la empresa tecnológica *ThoughtWorks* dos veces al año en el que se listan varias plataformas, técnicas, herramientas, lenguajes y frameworks que pueden tener un impacto importante en la industria.

La idea principal detrás del micro-frontend es aplicar el concepto de los microservicios al frontend. En vez de entender las vistas de una web como un único elemento, se dividen en un conjunto de funcionalidades suficientemente complejas como para ser gestionadas por equipos de desarrollo distintos.

● Directrices para diseñar una arquitectura de micro-frontends.

Para diseñar una arquitectura de este tipo, hay varias decisiones importantes que tomar:

Definición

¿Cómo se dividirán las vistas? Existen dos formas principales de hacerlo: la división horizontal y la vertical. La horizontal se refiere a separar cada componente de la vista (la cabecera, el pie de página, un contenedor concreto…) y asignarlo a un equipo, que se coordinará con los demás para formar el resultado final, mientras que la vertical implica asignar cada vista concreta, con todos sus componentes a un equipo.

Composición

¿Dónde se combinan las diferentes vistas? Puede darse en el cliente, en el servidor o mediante *Edge Computing*.
En el caso de aplicaciones de página única la opción más eficiente es hacerlo en el cliente.

Enrutamiento

Al haber diferentes microaplicaciones, cada una maneja sus propias rutas, entonces, ¿dónde y cómo se realizará la lógica de enrutamiento? . También puede darse en el cliente, el servidor o mediante *Edge Computing*, aunque en este caso es posible utilizar más de uno. Generalmente se recomienda utilizar el mismo método que en la composición.

En cuanto al cómo, una posible solución sería implementar un enrutador general que tenga el control sobre todas las vistas, aunque esto incrementa el acoplamiento al ser un código compartido entre todos los equipos.

Enfoques de integración

La definición explicada hasta ahora comprende muchos enfoques que podrían ser considerados micro frontends. Un aspecto común a todos ellos es la existencia de una micro interfaz para cada página y una página general que aglutina aspectos utilizados en todas ellas (como el encabezado o el pie), aborda preocupaciones comunes, orquesta las diferentes micro interfaces. Algunos de estos enfoques:

- Composición de la plantilla del lado del servidor
- Integración en tiempo de compilación
- Integración en tiempo de ejecución a través de Iframes
- Integración en tiempo de ejecución a través de JavaScript

Estilismo

Las características del lenguaje css como son la herencia y la cascada hacen que se generen conflictos sobre todo cuando se integran estilos desarrollados por diferentes equipos.

Estos problemas se resuelven aplicando diversas soluciones que tienen como objetivo garantizar que los estilos escritos se comportarán de forma predecible al juntarse e integrarse.

Comunicación entre aplicaciones

Como primera recomendación es intentar que estos se comuniquen lo menos posible ya que establecer demasiadas comunicaciones reintroducen la clase de acoplamiento que se pretende evitar. Sea cual sea el enfoque de comunicación elegido, se trata de que los micro frontends se envíen mensajes o eventos entre sí evitando mantener un estado compartido.

- Test

  Respecto a las pruebas no hay mucha diferencia entre interfaces monolíticas y las micro interfaces. Normalmente las estrategías que se usan para probar interfaces monolíticas valen para reproducirse en cada interfaz individual.

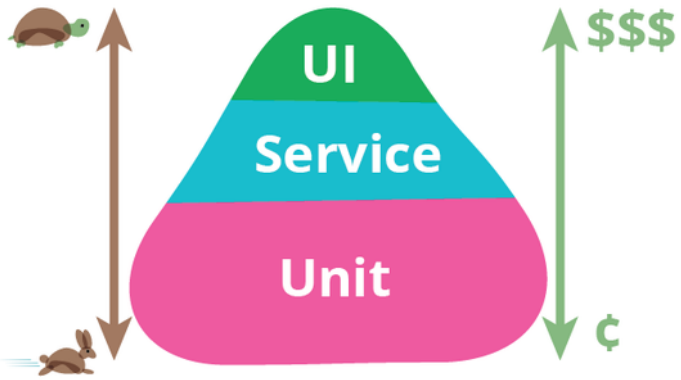
  Cada micro Interfaz individual tendrá su propio conjunto de pruebas que garantice la calidad del código.

El problema está a la hora de probar la integración de las diversas micro interfaces con la aplicación contenedora

Las pruebas funcionales solo deben cubrir aspectos que
no se pueden probar en un nivel inferior de la pirámide de prueba. Es decir:

Utilizar pruebas unitarias para cubrir su lógica comercial de bajo nivel y lógica de representación.
Utilizar pruebas funcionales sólo para validar que la página se encuentra correctamente ensamblada.
P.e: Cargar aplicación desde una url y que el título codificado en la micro interfaz esté presente en la página.

Objetivo principal: Validar la integración de las interfaces antes que la lógica interna de cada micro interfaz.



- # Ejemplo práctico

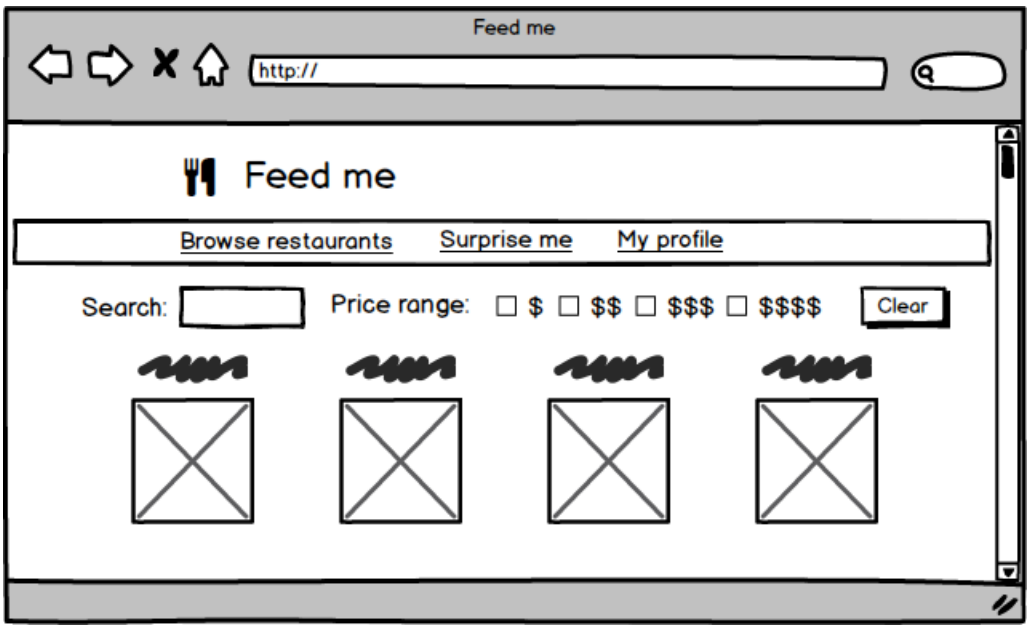
Sitio web donde los clientes pueden pedir comida a domicilio.

Debe haber:

- Página donde los clientes puedan navegar y buscar restaurantes. Los restaurantes deben poder buscarse y filtrarse por distintos atributos como: precio, tipo de comida etc.

- Cada restaurante necesitará su propia página que muestre su menú y permita que el cliente pueda elegir lo que quiere comer, así como aplicar descuentos u ofertas.

- Por otro lado, el usuario de la aplicación también podrá tener su propia página de perfil, con opiniones, restaurantes visitados, sus opciones de pago etc.



Toda esta funcionalidad perfectamente se podría dividir en distintos equipos de trabajo que trabajasen de manera independiente y finalmente, en conjunto hiciesen que todo funcionase. El equipo del perfil de usuario, no se preocupará en absoluto del menú de la página de los restaurantes por ejemplo.

- Ventajas y desventajas

Ventajas

1. Actualizaciones incrementales

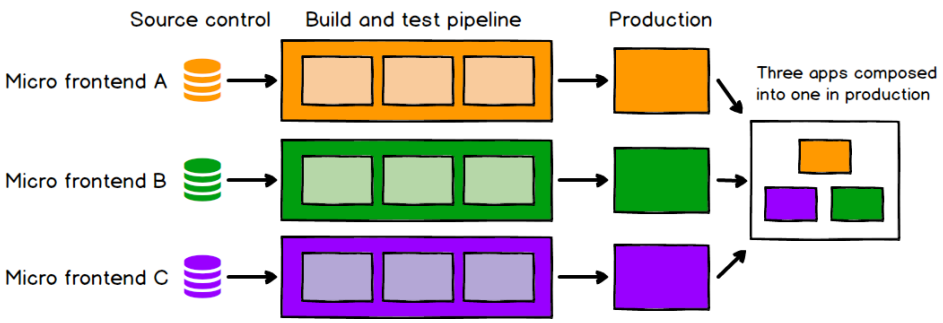
   Existe más libertad para tomar decisiones sobre partes individuales de nuestro producto y para hacer actualizaciones incrementales. Si hay un cambio importante en nuestro marco principal, cada micro interfaz se puede actualizar cuando tenga sentido, en lugar de parar y modificar todo en conjunto.

2. Bases de código simples y desacopladas

   Por definición el código fuente de cada micro interfaz individual será más pequeño que el de una única interfaz monolítica. Al ser más pequeño, también será más simple y fácil de trabajar.

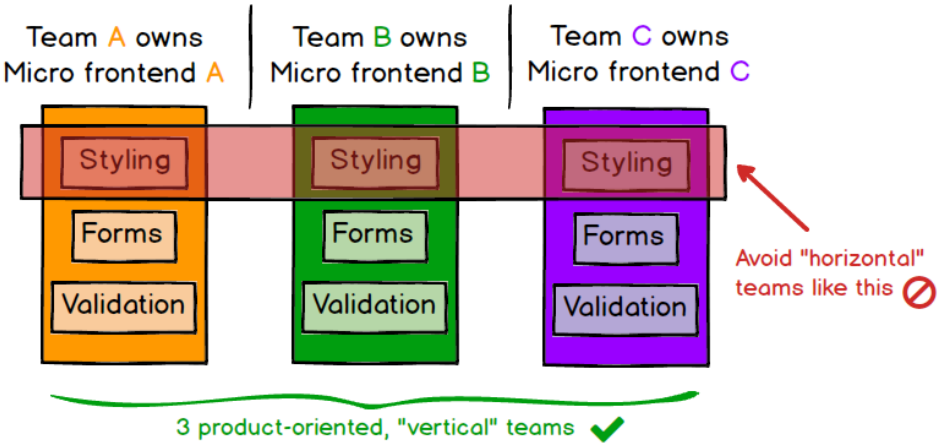
3. Despliegue independiente

Cada micro frontend se implementa en producción de forma independiente. Es decir, se implementan sin depender de otros. Formando finalmente un conjunto en el despliegue. Esto hace que si alguno falla, no se arrastre el fallo a otros.



4. Equipos autónomos

Cada aplicación ha de ser propiedad de un sólo equipo. Se suele dividir el producto en función de lo que verán los usuarios finales, de este modo cada micro frontend encapsulará solo una página de la aplicación, y será propiedad de un sólo equipo de extremo a extremo. Favoreciendo así la cohesión de equipos.



Desventajas

1. Tamaño de la carga útil

   Los paquetes de JS creados de forma independiente pueden provocar la duplicación de dependencias comunes, aumentando así el número de bytes que hay que enviar en la red a los usuarios finales. P.e: Si cada micro interfaz tiene su propia copia de react, estamos obligando a descargar react 5 veces.

2. Diferencias ambientales

Existen riesgos asociados si se desarrolla en un entorno que se comporte de manera diferente al de producción. Si nuestro contenedor de tiempo de desarrollo se comporta de manera diferente al de producción, podemos ver que nuestro micro frontend está roto o se comporta de manera diferente. Por lo tanto, la ventaja de la independencia de ejecutar el micro frontend en un modo “independiente” en lugar de la aplicación contenedora que lo aloja puede resultar peligroso.

3. Complejidad operativa

   Tener una arquitectura con micro frontends implica tener más cosas que administrar: más herramientas, más servidores, repositorios etc. Por lo que, es necesario reflexionar si se tiene la suficiente madurez técnica y organizativa para adoptar esta arquitectura sin sembrar un gran caos.

● Conclusión

Con el paso de los años el código frontend va volviéndose más complejo y surge la necesidad de llevar a cabo arquitecturas más escalables. Cuando se eligen micro frontends, por definición, se está optando por crear muchas cosas pequeñas en lugar de una cosa grande que trabajen en conjunto. Por ello, deberíamos poder escalar la entrega de software a través de equipos independientes y autónomos.
Pudiendo aportar algunas de las ventajas ya mencionadas, y a su vez también algunos inconvenientes. Por ello, dependiendo del tipo de sistema que se quiera diseñar, puede ser una buena opción a tener en cuenta en el mundo de la Ingeniería del Software.

● Bibliografía

Tanto información como imágenes obtenidos de las siguientes fuentes:

- Mezzalira,L.(22 de diciembre de 2019). *Micro-frontends decisions framework*. https://medium.com/@lucamezzalira/micro-frontends-decisions-framework-ebcd22256513
- Jackson, C.(19 de junio de 2019). *Micro frontends*. https://www.martinfowler.com/articles/micro-frontends.html