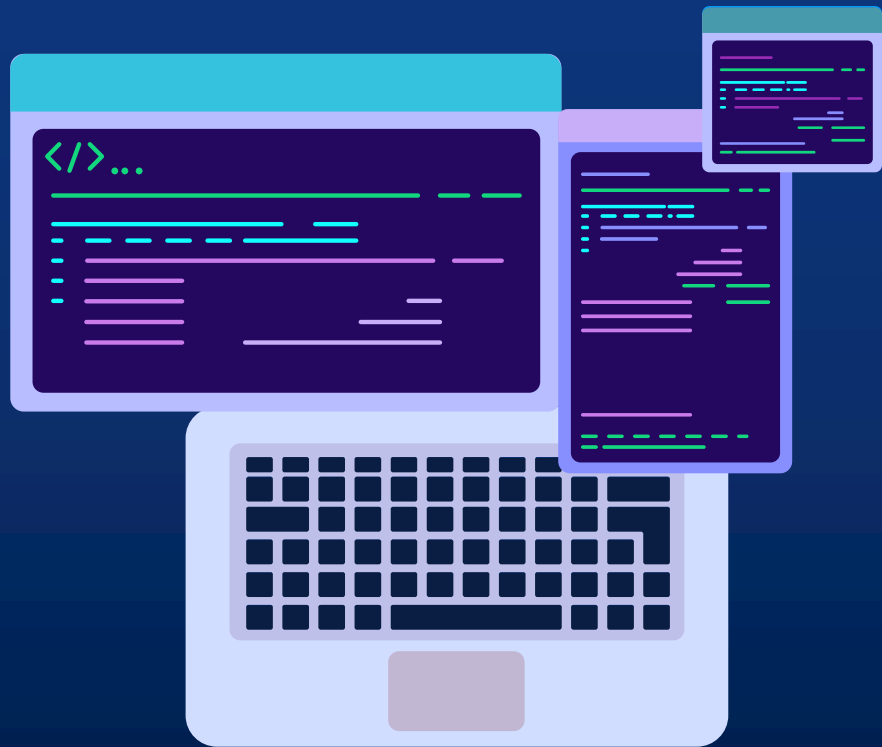


# Arquitectura Web basada en Microfrontends

Sebastian Alexander Diaz Paz



# Tabla de Contenido

01

## Microfrontends

Acercamiento teórico de la propuesta arquitectónica

02

## Tipos de implementación

Vertical / Horizontal

03

## Angular y Module Federation

Implementación tecnológica

04

## Aplicando "capa"

Evolucionando apps existentes

05

## Módulos dinámicos dinámicos


Desacoplamiento "total"

06

## Otras consideraciones


Para tener en cuenta





“Si podemos desacoplar el mundo  
backend en microservicios.  
¿Por qué no hacerlo en el frontend?.”

— Un desarrollador del común haciendo soporte en una web  
de legado.





01

## Microfrontends



# PROBLEMA DE LA CAJA DE PANDORA WEB



## PROBLEMA

A medida que los usuarios crecen, las necesidades del negocio crecen, los figmas son más y más granes...  
El repositorio web es **más complejo** de lo que se pensó en un inicio.



## PROPUESTA

Rescatando la propuesta funcional del mundo backend de separar por dominios funcionales independientes...  
¿Cómo **podemos aplicarlos** al frontend?



# VENTAJAS DE LOS MICROSERVICIOS

## AUTONOMOS

Dominio de negocio independiente

## ESPECIALIZADOS

Principio de responsabilidad única

## FACILMENTE ESCALABLES

Satisfacer la alta demanda en segundos

## REUSABLES

Aplicables en diversas soluciones

## AGILIDAD

Despliegues más frecuentes

## IMPLEMENTACIÓN SENCILLA

Ciclo de desarrollo estándar y simple

## AGNOTSTICOS

Stacks tecnológicos diferentes

## Resistencia

Puntos de fallo independientes





# VENTAJAS DE LOS ~~MICROSERVICIOS~~ MICROFRONTS

## AUTONOMOS

Dominio de negocio independiente

## ESPECIALIZADOS

Principio de responsabilidad única

## FACILMENTE ESCALABLES

Satisfacer la alta demanda en segundos

## REUSABLES

Aplicables en diversas soluciones

## AGILIDAD

Despliegues más frecuentes

## IMPLEMENTACIÓN SENCILLA

Ciclo de desarrollo estándar y simple

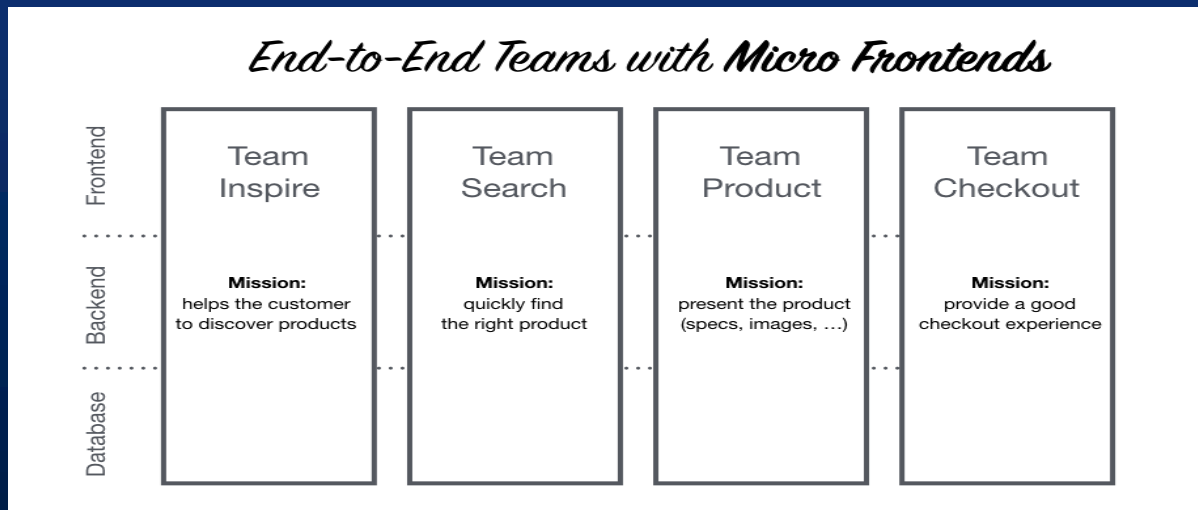
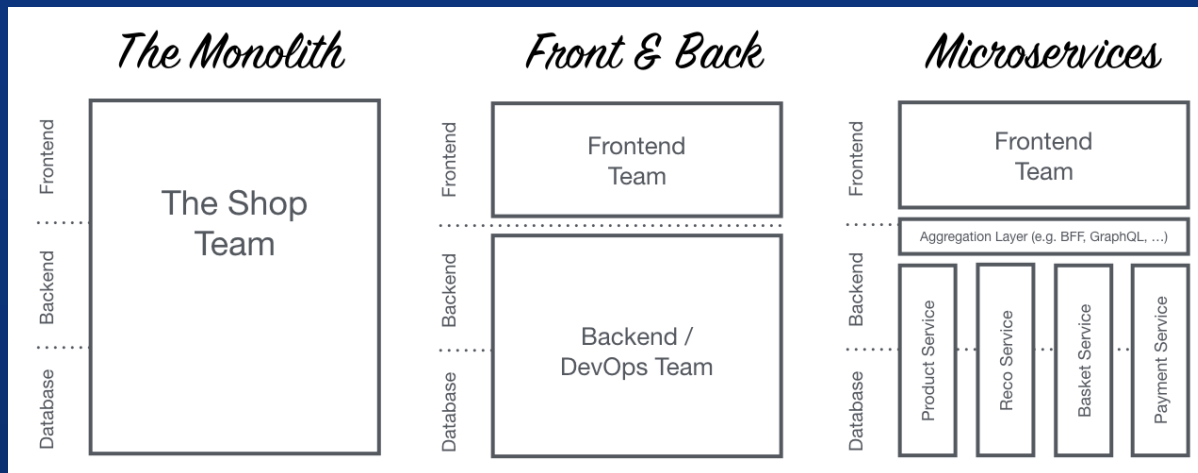
## AGNOTSTICOS

Stacks tecnológicos diferentes

## Resistencia

Puntos de fallo independientes









# 02

## Tipos de implementación





# EFOQUE DE IMPLEMENTACIÓN

## VERTICAL

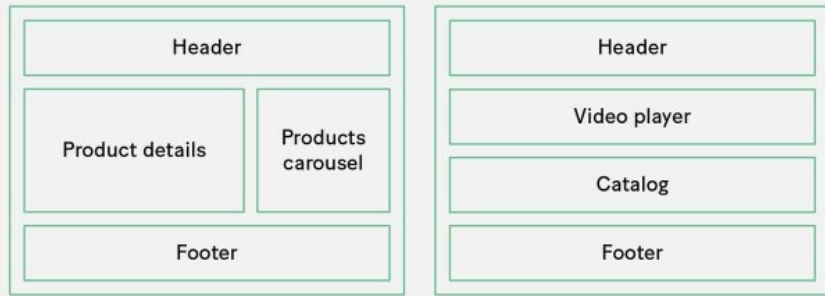
Mayor enfoque en el dominio del negocio y cada una de sus características

## HORIZONTAL

Mayor enfoque en la división de componentes y solventar necesidades complejas acopladas



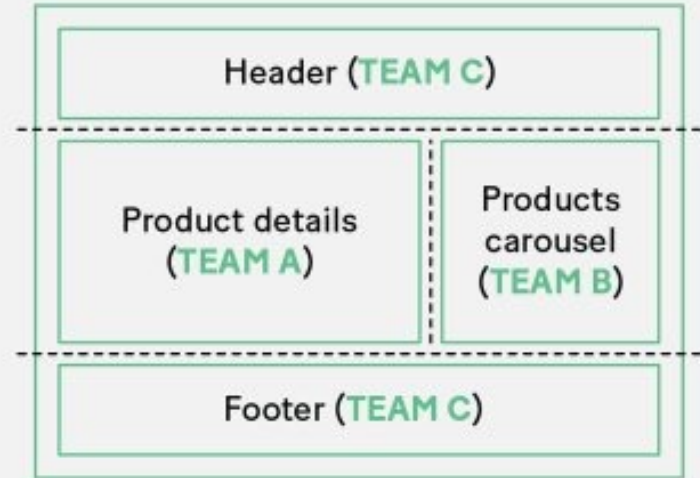
# VERTICAL



VERTICAL SPLIT

<https://increment.com/frontend/micro-frontends-in-context/>

# HORIZONTAL



HORIZONTAL SPLIT

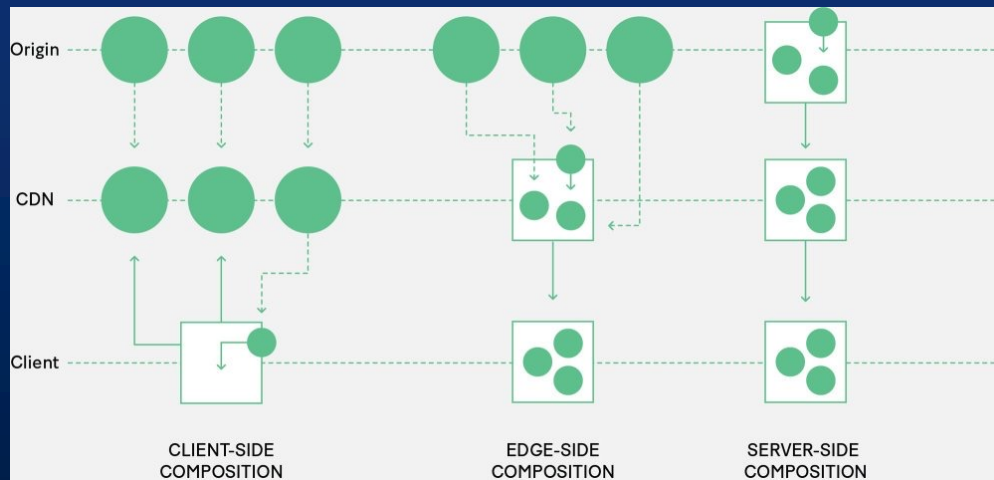
<https://increment.com/frontend/micro-frontends-in-context/>

# COMPOSICIÓN

## Creación y Renderización

Existen 3 tipos de composición:

- **Client-side.** (Más común)
- Edge-side. (Más versátil)
- Server-side. (Más eficiente)





# SOLUCIONES TECNOLÓGICAS

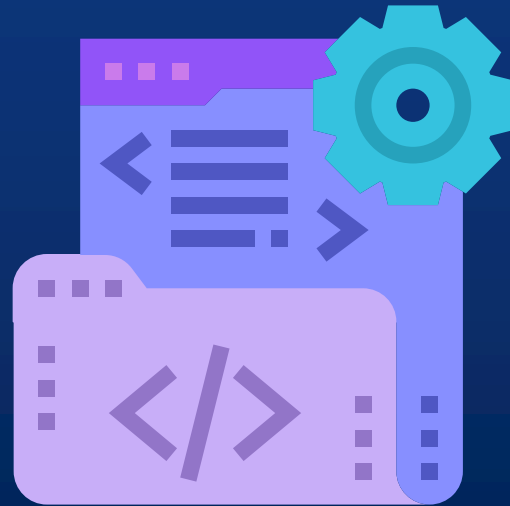
Herramientas y Utilidades

- Custom Elements. (Horizontal – Multi-Framework)
- Single SPA. (Vertical – Multi-Framework)
- **Module Federation.** (Vertical – “Multi”-Framework)



# 03

## Angular y Module Federation





# INTEGRANDO MODULOS EN RUNTIME

## **Webpack 5**

Desde la versión 5 del famoso bundler se incluyó una utilidad que permite la carga e importación de módulos desplegados y “compilados” desde diferentes fuentes.

## **Angular Architects - Module Federation**

El equipo de Angular Architects (Michael, Rainer y Manfred) desarrollaron un paquete npm que provee un custom bundler que permite además de “compilar” las aplicaciones Angular, generar archivos estáticos listos para ser importados en runtime por webpack.



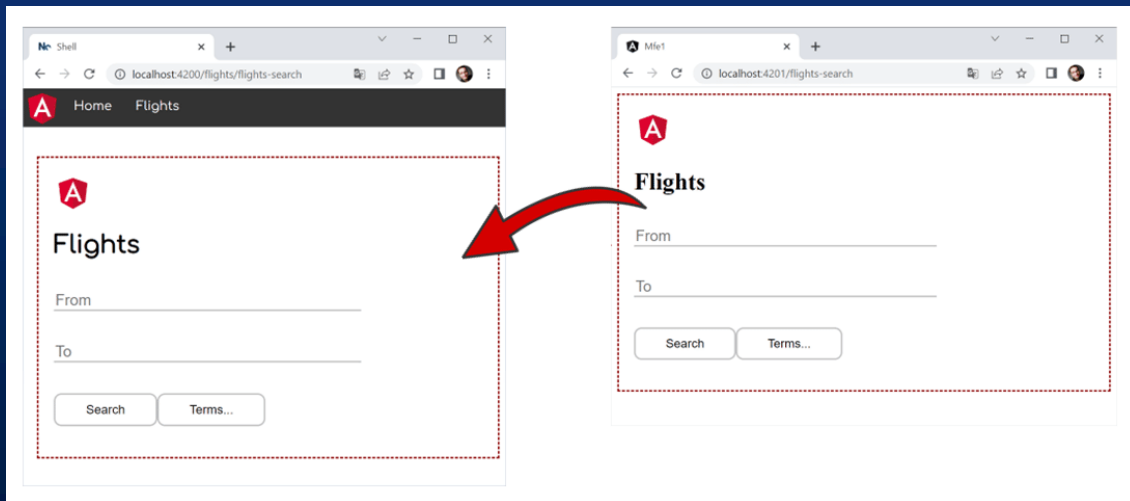
# COMANDOS

## Agregar Host

\$ ng add @angular-architects/module-federation  
--project shell --port 4200 --type host

## Agregar Remote

\$ ng add @angular-architects/module-federation  
--project mfe1 --port 4201 --type remote





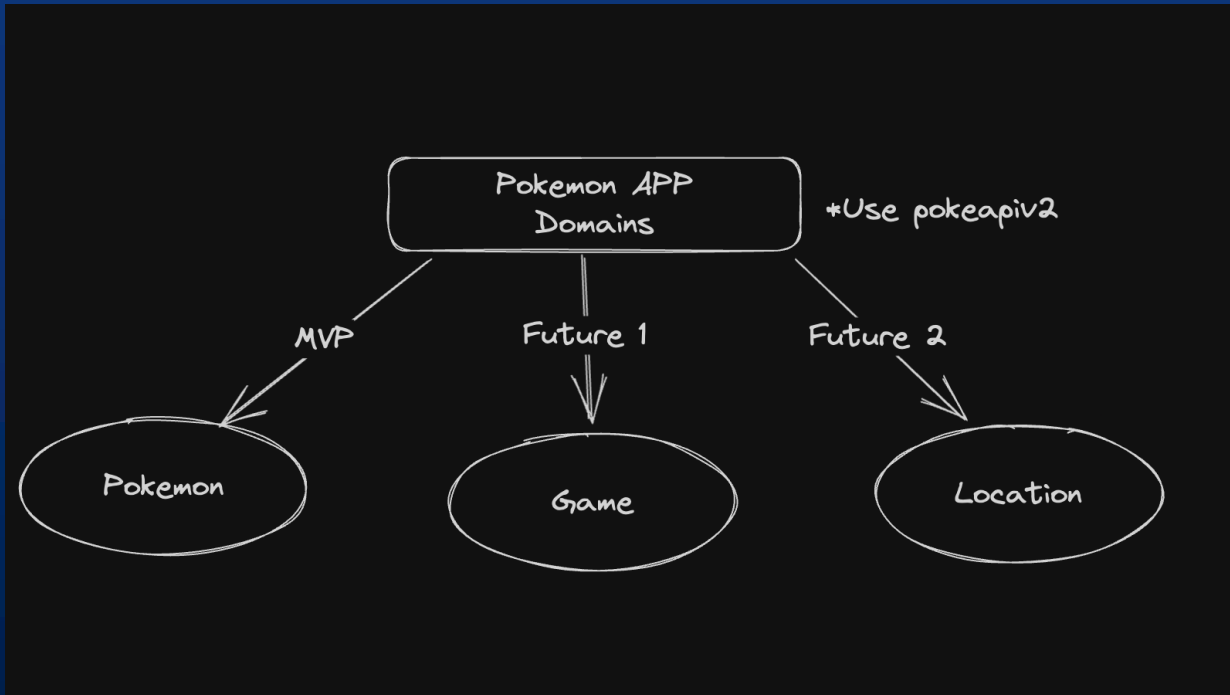


# 04

Aplicando "capa"

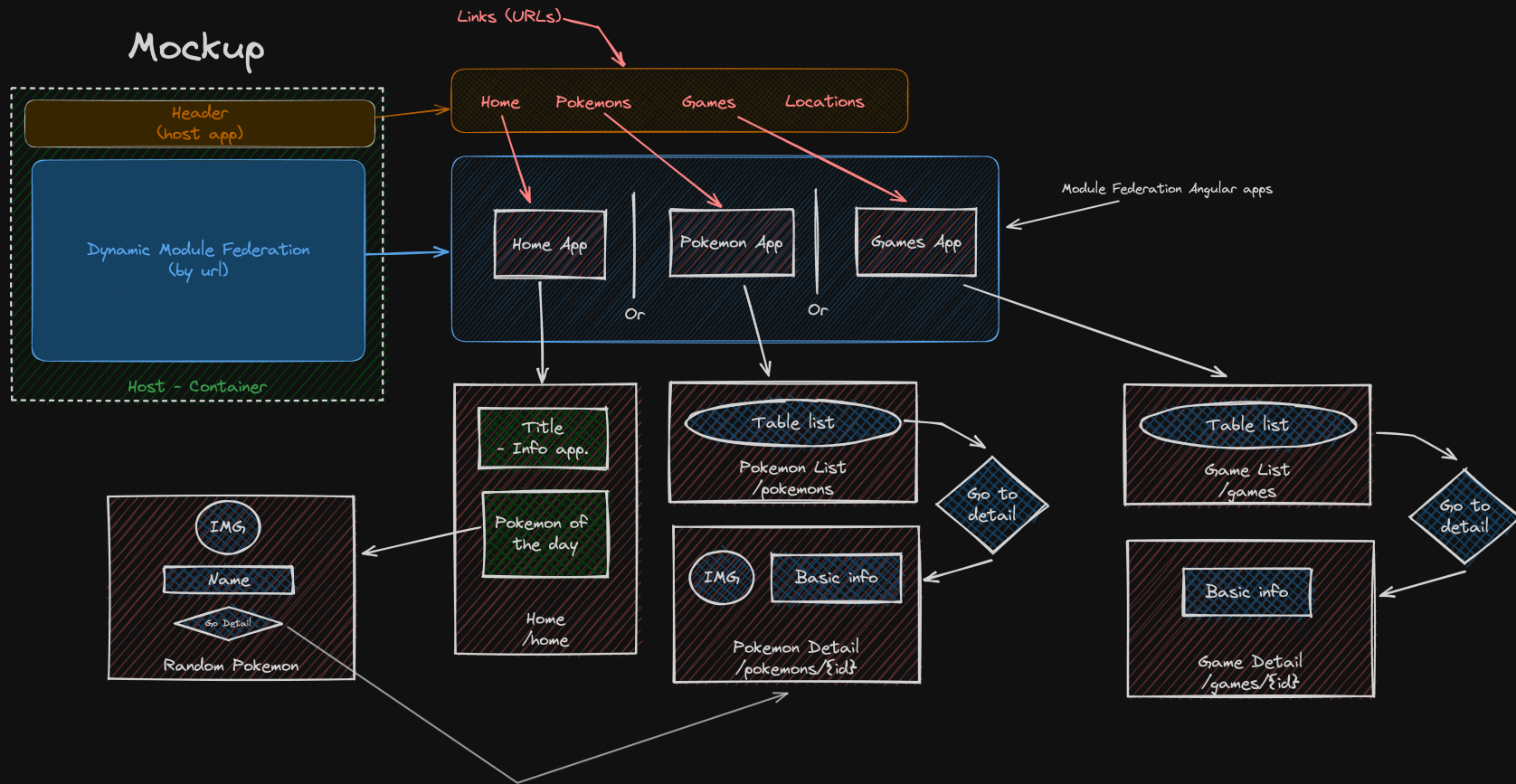


# MODELOS - DOMINIOS



# Poke-Summary APP

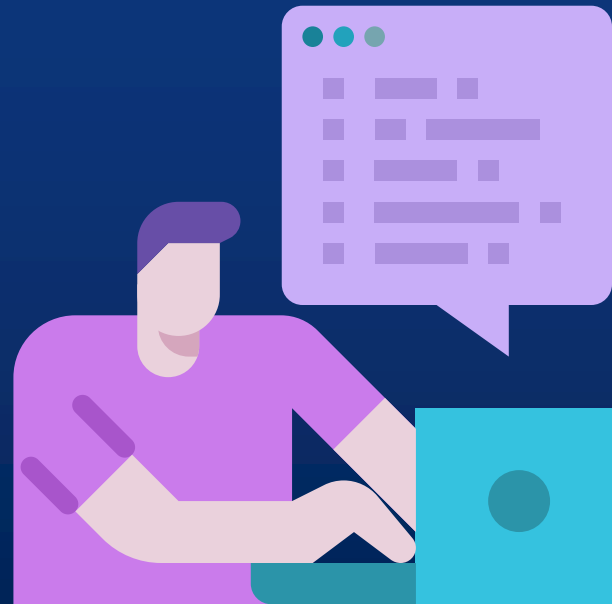
## Mockup





# 05

Módulos  
dinámicos  
dinámicos





# 06

## Otras consideraciones

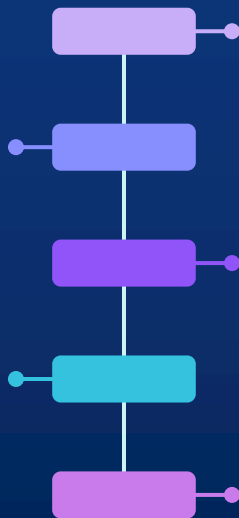


## Stack tecnológico - Necesidad

No existe la mejor implementación, depende de las necesidades.

## Compartir datos entre microfronts

Si bien, se puede considerar anti-patrón, será necesario encontrar un medio de comunicación común. (Eventos, WebStorage, Bilbiotecas, etc)



## CORS

Si se usan diferentes servidores y dominios URL, garantizar consumo CORS.

## Definir muy bien los dominios y contextos

Una mala división puede agregar complejidad accidental.

## Sistema de Diseño

Es vital definir un sistema de diseño para mantener la uniformidad de una aplicación web.

# THANKS



Repo:

<https://github.com/Dark-Light-20/Pokemon-Summary-APP>



**CREDITS:** This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik**