ORANGE STACK

Foundation Front-End

O time de Foundation Front-End, assim como outras tribos, também faz parte da Orange Stack. Nosso principal objetivo é aumentar a velocidade do desenvolvedor para construir e ou evoluir uma aplicação. Tendo isto como meta, a Foundation Front-End irá construir e propor modelos e ferramentas que visam alcançar o objetivo proposto.

Objetivo: Definição da Stack - Orange Portal

Contexto

Neste documento será feita uma análise para definição dos frameworks/libs que serão utilizados para a construção do Orange Portal tendo em vista que iremos utilizar a abordagem de micro frontend. Nossa recomendação, assim como as motivações para se utilizar micro frontend na construção do portal da Orange Stack pode ser lida neste link Orange Foundation - micro frontend.

Possíveis abordagens

Com o uso de micro frontends existe a possibilidade ou não de se utilizar múltiplos frameworks na construção da aplicação. Neste documento iremos expor os prós e contras de cada uma dessas abordagens.

Vale a ressalva de que a utilização de um Design System não somente em aplicações que utilizam micro frontends visa prover consistência visual, padronização, agilidade na criação de novas funcionalidades, dentre outros. Para tanto, todas as abordagens abaixo utilizam Design System, porém algumas abordagens podem influenciar diretamente em como esse Design System é implementado e utilizado nas aplicações.

Múltiplos Frameworks

A utilização de diferentes frameworks em uma mesma aplicação é possível e nos trazem vantagens e desvantagens.

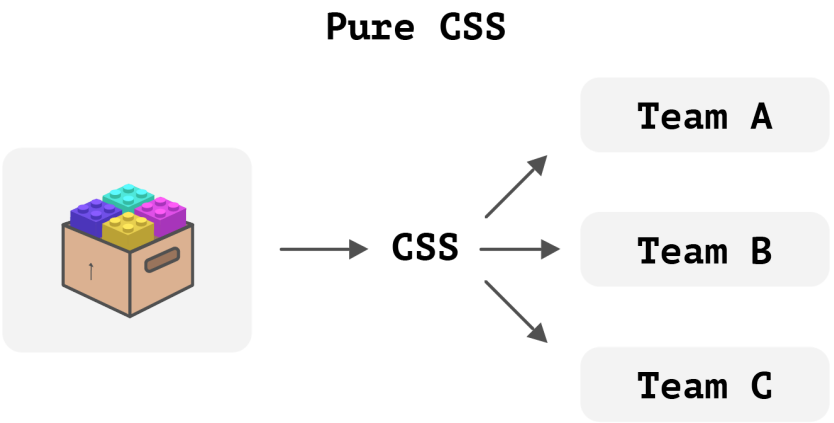
Vantagens

- Autonomia: Liberdade total para o time que está desenvolvendo a aplicação escolher o framework de sua preferência.
- Conhecimento especializado: Aproveitamento do conhecimento de desenvolvedores especialistas no framework escolhido.

Desvantagens

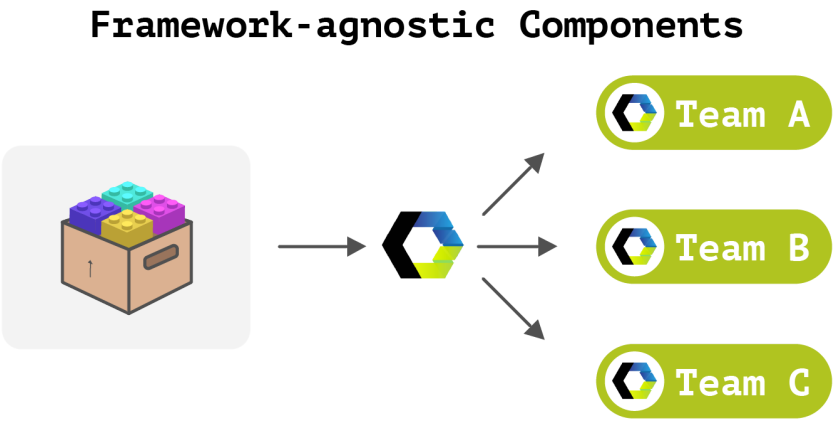
- Design System: A construção do Design System será altamente impactada, pois precisaremos fazer com que o mesmo Design System seja utilizado em diferentes frameworks. Para isto, basicamente podemos adotar três estratégias diferentes.

    CSS Puro: Neste modelo a estilização dos nossos componentes será feita por meio de classes CSS, um exemplo desta abordagem é o Twitter Bootstrap.



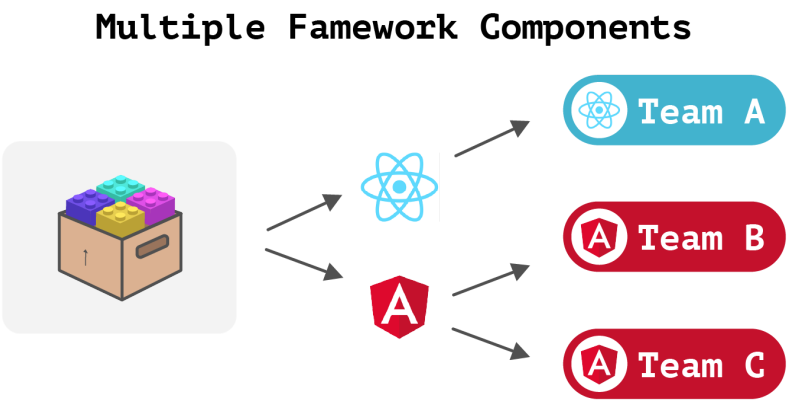
    Os problemas com esta abordagem é que nossos componentes seriam apenas estilos, não sendo possível adicionar comportamentos. Dado que os estilos se baseiam em markups e uma mudança no Design System, ou em alguma classe, impactaria diretamente todos os projetos que o utilizam e possivelmente necessitando de intervenção manual para adequação das novas classes.

Componentes agnósticos de Framework: Nesta abordagem, a construção dos componentes se dá por meio de Web Components. O grande ganho aqui é construir o componente apenas uma vez e reaproveitar entre os diferentes frameworks.



Os problemas desta abordagem começam na criação do componente. A construção de um DS utilizando apenas a API nativa não é muito encorajada, e por este motivo fez com que fossem criadas inúmeras bibliotecas para auxiliar na construção dos componentes, como por exemplo Stencil, Polymer, Fast, etc. Caso seja adotada a utilização de alguma dessas bibliotecas, um desenvolvedor necessitaria aprender uma nova linguagem para a construção de um componente. Tal escolha, poderia afetar negativamente o engajamento de contribuições no Design System. Outro problema desta abordagem é que a biblioteca do React até o presente momento (04/2021) não fornece total suporte para Web Components. Uma simples passagem de parâmetro do tipo objeto ou array utilizando Web Component, necessitaria de criar um wrapper adicional na aplicação React para que isto fosse possível. A utilização dos eventos (onClick, onChange, etc) também seriam diretamente impactados, sendo necessário também a utilização do wrapper para que funcione corretamente.

Múltiplos Design System: Nesta abordagem, a ideia é criar um design system específico para cada framework utilizado.



A maior desvantagem desta abordagem se dá na escalabilidade do Design System. Construir, manter e evoluir apenas um Design System já é algo desafiador e escalar estes problemas para a quantidade de diferentes frameworks em nossa aplicação se torna inviável.

- Performance: Dado que cada time tem a liberdade de escolher seu framework, não será incomum termos diferentes frameworks rodando em paralelo, inclusive um mesmo framework podendo ter mais de uma versão sendo carregada no Portal da Orange. O grande problema disto é que diferentes frameworks possuem seu próprio core, não sendo possível reaproveitá-los nem mesmo entre diferentes versões do mesmo framework. O usuário final iria precisar de uma espera muito maior para o carregamento inicial da aplicação, mesmo utilizando técnicas de carregamento em lazy e cache.
  Um outro exemplo seria o não reaproveitamento de libs específicas para algum framework. Vamos supor um cenário, onde um time que desenvolve em React necessite de criar em seu domínio uma tela com drag and drop e para isto, resolva utilizar alguma lib já existente, como por exemplo: https://github.com/atlassian/react-beautiful-dnd. Caso alguma outra equipe que utilize outro framework, por exemplo Angular, não poderia aproveitar desta biblioteca já utilizada e necessitaria de utilizar uma outra, aumentando o tamanho do bundle e podendo até mesmo ter comportamentos diferentes em locais diferentes do Portal Orange.

- Criação de nichos específicos de conhecimento: Ao dar liberdade de escolha para as squads na escolha do framework, elas podem optar por um framework que nenhuma outra squad tenha conhecimento. O que causaria problemas caso esta squad necessitasse de ajuda para desenvolvimento de novas funcionalidades ou até mesmo se os desenvolvedores frontend desta squad mudassem de projeto, pois não

teriam outros desenvolvedores aptos para atuarem com essa tecnologia e talvez seria necessário reescrever o módulo em outra linguagem.

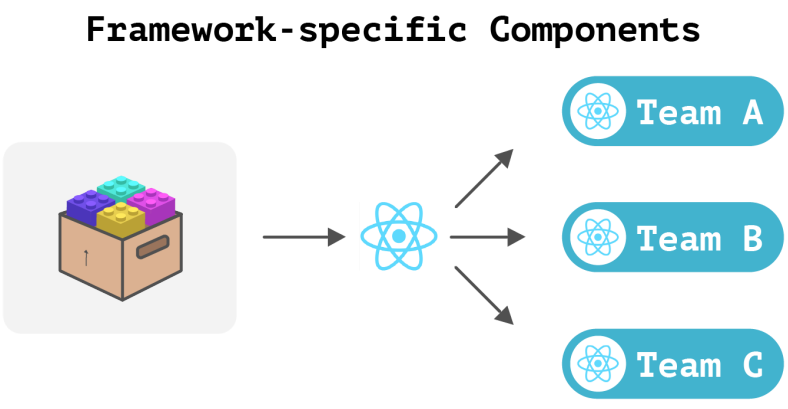
- Comunicação entre frameworks: Este desafio não se dá somente com a utilização de vários frameworks, mas é agravado por esta escolha. A comunicação entre os diferentes frameworks deve ser pensado para que haja interação entre os módulos. Para isto, algumas abordagens são possíveis, como por exemplo o uso de um EventBus entre as camadas. Em contrapartida, não seria possível utilizar funcionalidades já existentes caso estivéssemos utilizando um único framework. Exemplo: Se utilizássemos apenas Angular, poderíamos ter um Service compartilhando informações entre todos os outros módulos do Angular, o mesmo seria válido para React, onde poderíamos usar o Context para prover um contexto entre os diferentes módulos da aplicação.

Único Framework

A utilização de um único framework utilizando micro frontend também traz vantagens e desvantagens

Vantagens

- Design System: A construção do Design System é feita na linguagem do framework escolhido, de uma forma muito simples e não sendo necessário wrappers ou de outras bibliotecas para a construção do mesmo, o que proporciona um maior engajamento para utilização e contribuição do DS.



- Performance: O compartilhamento de bibliotecas, e do próprio core do framework é possível, diminuindo o tamanho do bundle e consequentemente melhorando o tempo de carregamento da aplicação.

- Comunicação entre módulos: É possível utilizar funcionalidades do próprio framework escolhido para fazer a comunicação entre os diferentes micro frontends.
- Compartilhamento de conhecimento e padronização: Utilizar um único framework também facilita a padronização do código, o compartilhamento de conhecimento entre os times e reduz o tempo de onboarding quando for necessário movimentações entre as squads.

Desvantagens

- Autonomia: Neste cenário não há autonomia para a escolha de um framework em que a squad tenha maior conhecimento, fazendo com que os desenvolvedores que não tenham aptidão no framework escolhido necessitem de treinamento, estudo e apoio para que consigam performar em alto nível.

Nossa recomendação

A escolha pela utilização de diferentes frameworks nos traz a possibilidade de desenvolver nosso micro frontend com a tecnologia em que temos mais conhecimento. Entretanto, se analisarmos todos os impactos que esta escolha traz no projeto como um todo, e que se encontram listados neste documento, tal abordagem pode causar mais problemas do que soluções. Dois dos pontos mais críticos que a utilização de micro frontend nos traz são performance e incompatibilidade visual. A escolha de se utilizar mais de um framework na mesma aplicação, pode agravar ainda mais esses problemas.

Para tanto, nossa recomendação como Foundation, na construção do Orange Portal é a utilização de um único framework.