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: Construção do portal Orange Stack

Contexto

Dado que a Orange Stack é um ecossistema de produtos e ferramentas, não é estranho imaginar que a construção deste portal irá envolver vários times diferentes, estamos falando de centenas de desenvolvedores construindo uma mesma plataforma. Times estes que poderão ser compostos de engenheiros de software do Itaú, Zup ou ambos.

Sabemos que o desenvolvimento Front-End Web no Itaú fomenta a adoção do framework Angular na criação de aplicações. Tal decisão é, sem sombra de dúvidas, uma escolha sábia dado o contexto, necessidades e suas motivações.

Por outro lado, o desenvolvimento Front-End Web na Zup, principalmente no time de produtos, em quase sua totalidade, se deu por meio de React. A escolha das tecnologias utilizadas na construção das aplicações também sempre foram pautadas por meio de reuniões entre especialistas, sempre tendo em vista os prós e contras das soluções existentes.

A nossa escolha por micro frontend

Sabendo que Angular e React são, atualmente, as duas libs mais utilizadas para a construção de SPA (Single Page Application) e que o portal da Orange Stack poderia ser desenvolvida utilizando estes frameworks, a equipe da Orange Foundation foi responsável em fazer uma

validação técnica a partir da experimentação com o intuito de definir qual ou quais stacks seriam utilizadas na construção do portal.

No contexto de Orange Stack, existem tribos com domínios muito bem definidos, como por exemplo Orange Pipes, Orange Data, Orange Analytics. Queremos portanto que cada uma dessas tribos possua autonomia para a construção do seu próprio front-end, tenha independência de deploys e consiga fazer o monitoramento de sua aplicação dentro do portal. Tendo isto como objetivo, o modelo de arquitetura em micro frontend se encaixa perfeitamente na extensibilidade, proporção e autonomia que queremos entregar.

Desafios de uma arquitetura em micro frontend

Dar este nível de autonomia para os times traz vários desafios, como por exemplo a integração entre as diferentes partes da Orange Stack. Não podemos dar autonomia e perder em **consistência e performance**.

Devemos ter consistência na experiência do usuário da Orange Stack, ou seja, independentemente de qual foi a escolha da equipe de desenvolvimento, devemos garantir que o usuário final não seja comprometido e que também a manutenção do nosso portal não seja prejudicada por esta decisão.

Nota: A arquitetura de micro frontends não resolve todos os problemas, de todas as aplicações. Cada aplicação possui seu contexto, suas particularidades e seus desafios. Micro frontend pode adicionar uma complexidade sem necessidade em seus projetos, pense bem ao adotar tal estratégia.

Modelos de micro frontend

A utilização de micro frontend possui várias vertentes, sendo possível serem implementadas de inúmeras formas diferentes além de ser possível, ou não, a utilização de vários frameworks em conjunto. Acreditamos que essas decisões devem ser avaliadas antes de cada construção de uma nova aplicação que decida implementar este modelo de arquitetura.

Para a construção do portal da Orange Stack, foram levantadas as principais abordagens para utilização de micro frontends e iremos detalhar as vantagens e desvantagens que encontramos em cada uma delas.

Múltiplos frameworks utilizando Single-SPA

A utilização do framework Single-SPA é uma das abordagens mais comuns quando o assunto é micro frontend. Tal abordagem é geralmente adotada pela abstração que o framework proporciona principalmente na convivência e comunicação entre os diferentes micro frontends.

Vantagens

- Framework conhecido na comunidade
- Abstração para utilização de múltiplos frameworks (Angular, React, Vue, etc)
- Deploy separados dos micro frontends
- Abstração no controle de rotas
- Dev tools (single-spa-inspector)

Desvantagens

- Recente suporte ao SSR.
- Necessidade de aprender um novo framework e como seu ciclo de vida funciona.
- Compartilhamento de dependências entre os frameworks não é otimizado
- Design System seria impactado. (Veja o detalhamento deste ponto acessando o six pages sobre a definição da Stack -Orange Foundation - Stack)

Múltiplos frameworks utilizando Proxy Reverso

A utilização de proxy reverso para o propósito de utilização dos micro frontends foi pensada para dar autonomia aos times. Tal abordagem se baseia que a orquestração das solicitações ao nosso domínio serão tratadas por uma lógica no servidor que fará o proxy das solicitações com base no path. Ou seja, a implementação neste ponto é mais uma abstração do lado do servidor para lidar com a composição dos micro frontends.

Vantagens

- Deploy independente
- Autonomia na escolha dos frameworks a serem utilizados
- Simplifica os problemas do CORS, uma vez que as solicitações de aplicativos estão todas sendo feitas para a mesma origem.

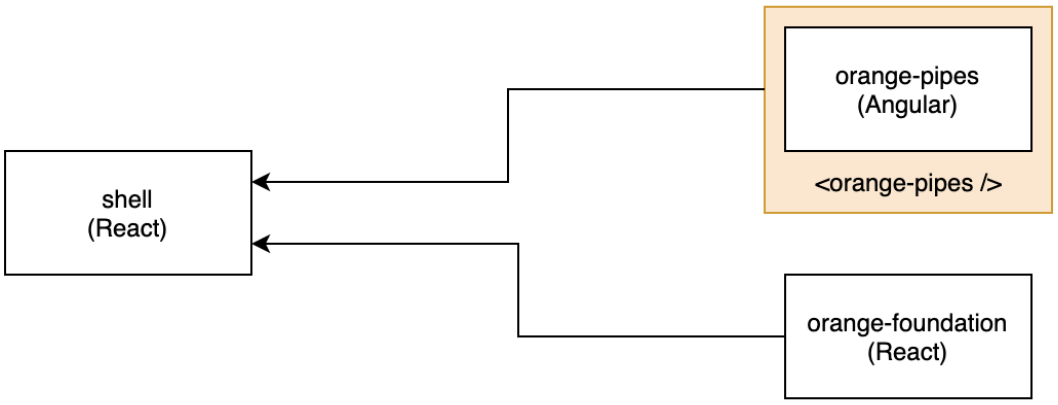
Desvantagens

- Esforço para construção do servidor de orquestração
- Comprometimento da experiência das SPA's
- Não possui abstração para comunicação entre os micro frontends, sendo necessário implementar ou escolher alguma lib para este propósito.
- Bundle duplicado entre os micro frontends, prejudicando a performance da aplicação final.
- Design System seria impactado. (Veja o detalhamento deste ponto acessando o six pages sobre a definição da Stack -Orange Foundation - Stack)

Múltiplos frameworks utilizando WebPack Module Federation e encapsulamento com Web Components.

Outra abordagem, para a construção de micro frontends é a utilização do Webpack Module Federation. Nesta abordagem, cada framework seria uma aplicação standalone, encapsulada como um web component provendo um entrypoint para que o shell da Orange Stack a carregasse sob demanda.

Criamos todo o ambiente com uma aplicação shell (React), e dois micro frontends, orange-pipes (Angular) e orange-foundation (React) conforme demonstrado no diagrama abaixo:



O repositório desta demonstração pode ser encontrado no repositório: https://github.com/pedronaveszup/pilha-laranja

Em cada micro frontend, seja ele em Angular ou React, foi necessário criar um encapsulamento de web component para utilização na aplicação shell, sendo que o web component poderia conter a aplicação inteira ou somente partes dela (widgets).

Para a comunicação entre os micro frontends foi criada uma biblioteca de EventBus onde é possível enviar e receber eventos por meio da API de CustomEvent() - Web APIs | MDN.

É possível termos um controle do estado e propriedades entre micro frontends, fazendo uso de uma lib para tal necessidade.

Vantagens

- Compartilhamento de libs (build-time + ModuleFederationPlugin), melhorando a performance da aplicação.
- Deploy independentes
- Possibilidade de se utilizar múltiplos frameworks
- Não é atrelado a nenhum framework
- Apesar do pouco tempo, muito bem visto na comunidade (https://github.com/module-federation/module-federation-examples) (https://github.com/manfredsteyer/multi-framework-micro-frontend)

Desvantagens

- Muito recente - Webpack 5 Release do dia 10/10/2020
- Não há evidências de uso em ambiente produtivo
- Não possui abstração para tratamento de rotas, ficando a cargo da aplicação shell (Angular Router / React Router)
- Não possui abstração para comunicação entre os micro frontends, sendo necessário implementar ou escolher alguma lib para este propósito.
- Design System seria impactado. (Veja o detalhamento deste ponto acessando o six pages sobre a definição da Stack -Orange Foundation - Stack)

Especificidade do React com a utilização do Web Components

- É necessário criar abstrações para o uso de WebComponents em React, visto que o React não tem um suporte muito bom para Web Components. (Veja o detalhamento deste ponto acessando o six pages sobre a definição da Stack -Orange Foundation - Stack)

Especificidade do Angular com a utilização do Webpack Module Federation:

- É necessário utilizar yarn para substituir o webpack para a versão 5 no Angular.

Único framework com Webpack Module Federation

Também utilizando Webpack Module Federation porém, com uma abordagem que visa melhorar a manutenibilidade e a experiência de desenvolvimento. Nessa abordagem é definido um framework único para desenvolvimento da aplicação Shell e das demais aplicações.

Vantagens

- Facilidade no roteamento entre os micro frontend
- Bundle mais enxuto, pois as libs em comum são carregadas apenas uma vez e gerenciada pelo Webpack
- Aplicação Shell menos complexa e com melhor manutenibilidade
- Design System na lib/ framework escolhido - sem a necessidade de web component na construção do Design System
- Transição de pessoas entre as Squads, pois a Stack será a mesma
- Deploy independentes
- Apesar do pouco tempo, muito bem visto na comunidade (https://github.com/module-federation/module-federation-examples) (https://github.com/manfredsteyer/multi-framework-micro-frontend)

Desvantagens

- Muito recente - Webpack 5 Release do dia 10/10/2020
- Não há evidências de uso em ambiente produtivo
- Perda da autonomia das squads para escolherem o framework

Nossas escolhas

A adoção de um único Framework como Stack de desenvolvimento garante um controle maior da performance da aplicação, pois ela se limita ao entendimento e expertise de um único framework e também reduz o tamanho do bundle, por ser necessário baixar o core de apenas um framework. Há também uma melhor manutenibilidade da aplicação, com um gerenciamento melhor das rotas entre micro frontends e criação de um Design System na mesma linguagem do Framework

escolhido. Além de eliminar a necessidade de wrappers para integração entre plataformas e remover a necessidade de uma lib para a comunicação entre diferentes frameworks.

Essa abordagem também permite a integração e movimentação entre os times, pois, com todos utilizando o mesmo framework, é mais fácil criar padrões de desenvolvimento e, assim, reduzir o tempo de aprendizado quando houver movimentação entre as diferentes Squads.

Tendo em vista estes prós e ainda sim visando garantir a autonomia das tribos no desenvolvimento de suas aplicações, acreditamos que o modelo a ser seguido na construção da Orange Stack deva ser utilizando **um único framework com Webpack Module Federation**.

Analisamos a utilização dos dois principais frameworks atuais para a construção do portal Orange Stack, sendo eles Angular e React. Esta escolha se deu devido a ampla utilização dos mesmos em grandes cases do mercado, a curva crescente de procura por desenvolvedores em ambos e a evolução de cada framework diante das necessidades que temos no desenvolvimento web hoje em dia em projetos de médio e grande porte.

Ambos são altamente eficazes para se trabalhar com micro frontends utilizando Webpack Module Federation, sendo uma particularidade do Angular a necessidade de se utilizar yarn para sobrepor a lib do webpack para a versão 5 (versão esta que contempla a utilização do Module Federation). Vale a ressalva de que não achamos que esta necessidade particular do Angular foi determinante na escolha ou não deste framework. Nossa escolha foi baseada principalmente na análise dos times que sabemos que estão atuando no desenvolvimento da Orange Stack, como por exemplo a tribo da Orange Hub e Orange Frontend no âmbito da geração de templates. As duas tribos possuem maiores conhecimentos específicos na lib React. Tendo isto em vista, e todos os outros pontos supracitados, escolhemos **React** como lib para toda a construção da Orange Stack.

Sabemos que as escolhas definidas pela Foundation serão refletidas nas demais tribos, inclusive na criação do Design System da Orange Stack e nossa missão como Foundation é fornecer todo o apoio necessário na utilização desta stack fornecendo templates, padronizações e ferramentas, garantindo assim que a Orange Stack tenha conformidade em seu desenvolvimento.