Entendendo os anti-patterns na arquitetura de Micro-Frontends

Pedro Henrique Oliveira

Solutions Architect
Amazon Web Services

Ricardo Tasso

Partner Solutions Architect
Amazon Web Services

"There is no compression algorithm for experience"

Andy Jassy – CEO of Amazon

Micro-Frontends são a ***representação técnica de um sub-domínio de negócio***, eles permitem ***implementações independentes*** com a ***mesma ou diferente tecnologia.***

E então, eles podem ***minimizar o código compartilhado*** com outros sub-domínios e eles são ***propriedade de um único time.***

Micro-Frontends são a ***representação técnica de um sub-domínio de negócio***, eles permitem ***implementações independentes*** com a ***mesma ou diferente tecnologia.***

E então, eles podem ***minimizar o código compartilhado*** com outros sub-domínios e eles são ***propriedade de um único time.***

Micro-Frontends são a ***representação técnica de um sub-domínio de negócio***, eles permitem ***implementações independentes*** com a ***mesma ou diferente tecnologia.***

E então, eles podem ***minimizar o código compartilhado*** com outros sub-domínios e eles são ***propriedade de um único time.***

Micro-Frontends são a ***representação técnica de um sub-domínio de negócio***, eles permitem ***implementações independentes*** com a ***mesma ou diferente tecnologia.***

E então, eles podem ***minimizar o código compartilhado*** com outros sub-domínios e eles são ***propriedade de um único time.***

Micro-Frontends são a ***representação técnica de um sub-domínio de negócio***, eles permitem ***implementações independentes*** com a ***mesma ou diferente tecnologia.***

E então, eles podem ***minimizar o código compartilhado*** com outros sub-domínios e eles são ***propriedade de um único time.***

Micro-Frontends são a ***representação técnica de um sub-domínio de negócio***, eles permitem ***implementações independentes*** com a ***mesma ou diferente tecnologia.***

E então, eles podem ***minimizar o código compartilhado*** com outros sub-domínios e eles são ***propriedade de um único time***.

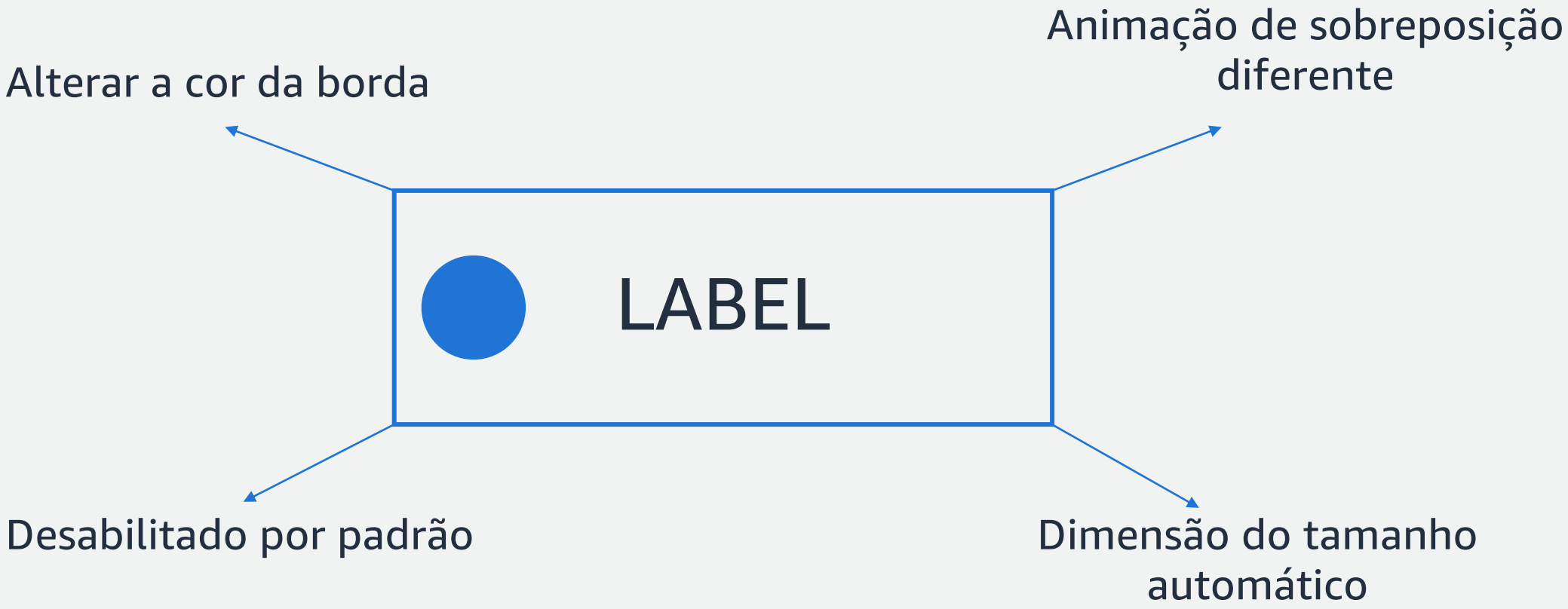
Benefícios dos Micro-Frontends

1. Upgrades incrementais
2. Descentralização
3. Redução do “Team cognitive load”
4. Dimensione a tecnologia bem como a organização

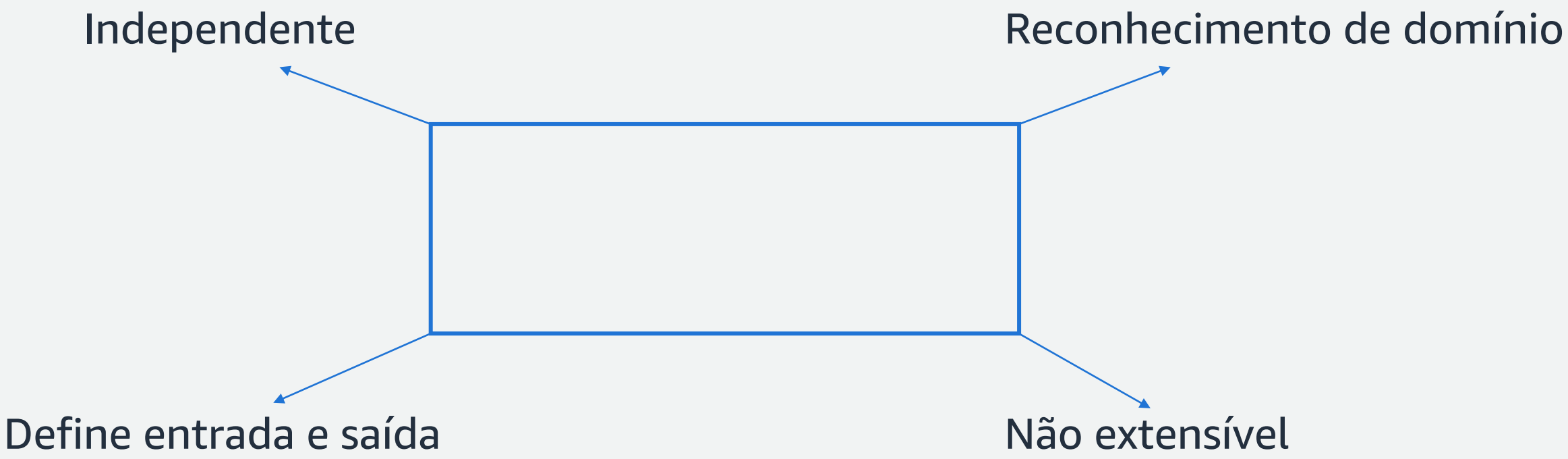
“*Yin and Yang*”

(Micro-Frontends e Componentes)

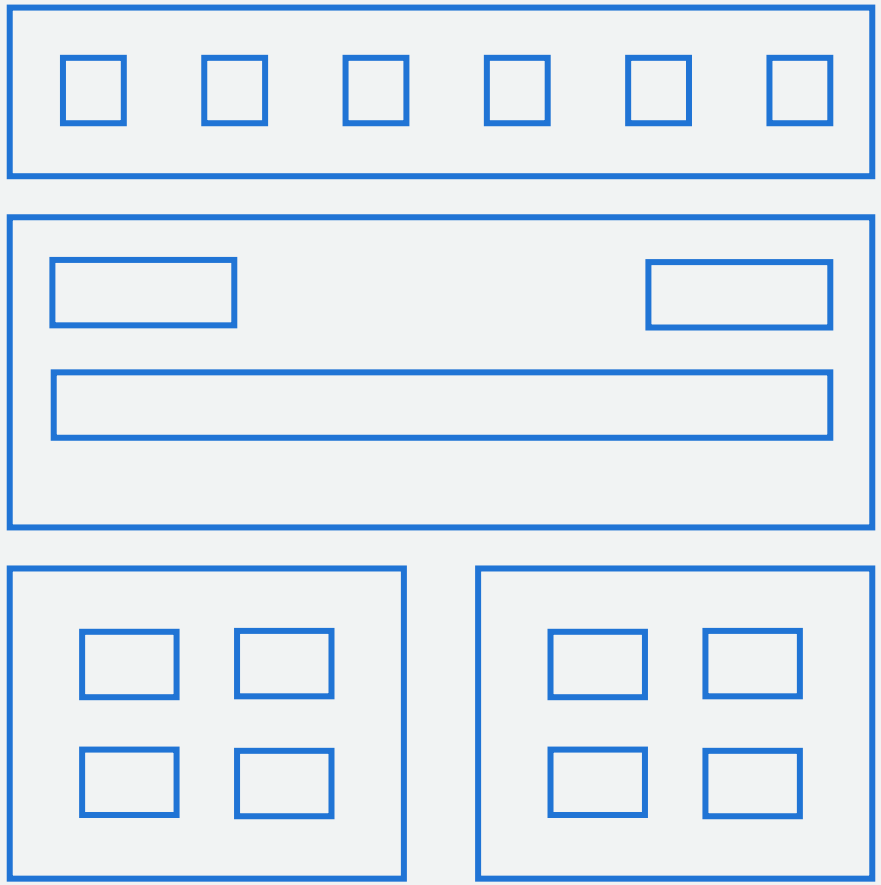
Um componente



Um Micro-frontend



Muitos... Componentes?

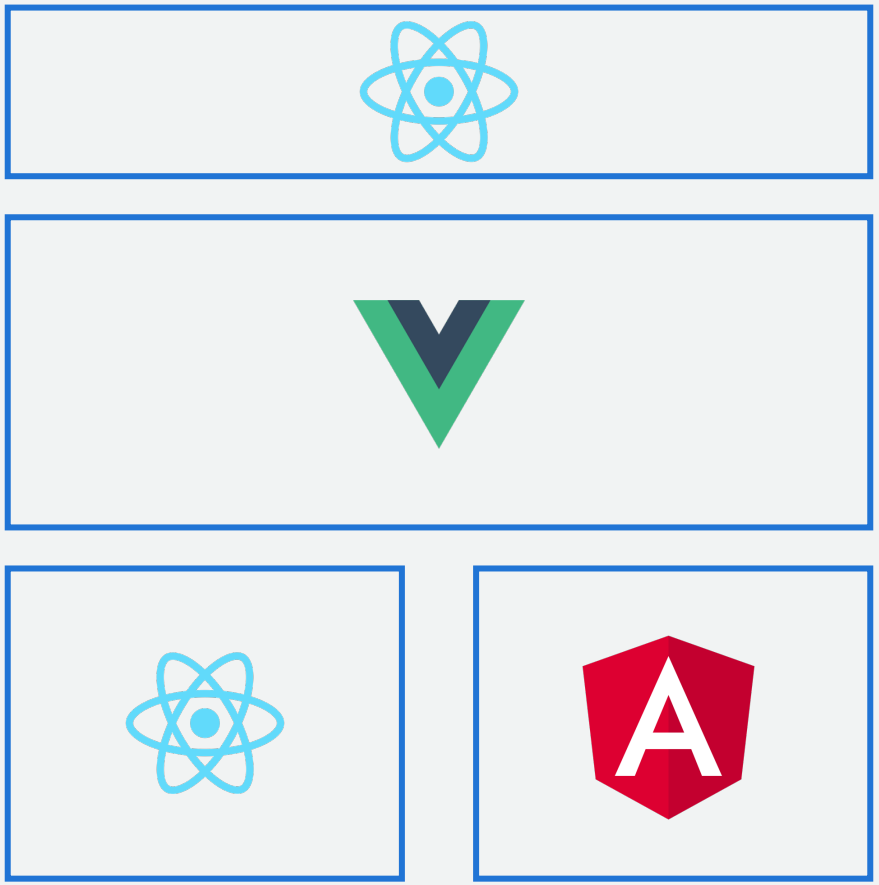


Você está projetando um micro-frontend ou um componente?

“Hydra of Lerna”

(Abordagem multi-frameworks)

Frameworks em todos os lugares!



Quantas
Bibliotecas **ou** *frameworks de UI*
você usaria em um SPA?

Abordagem Multi-framework

1 Lidando com Sistemas Legados

2 Migrando para um novo framework/biblioteca

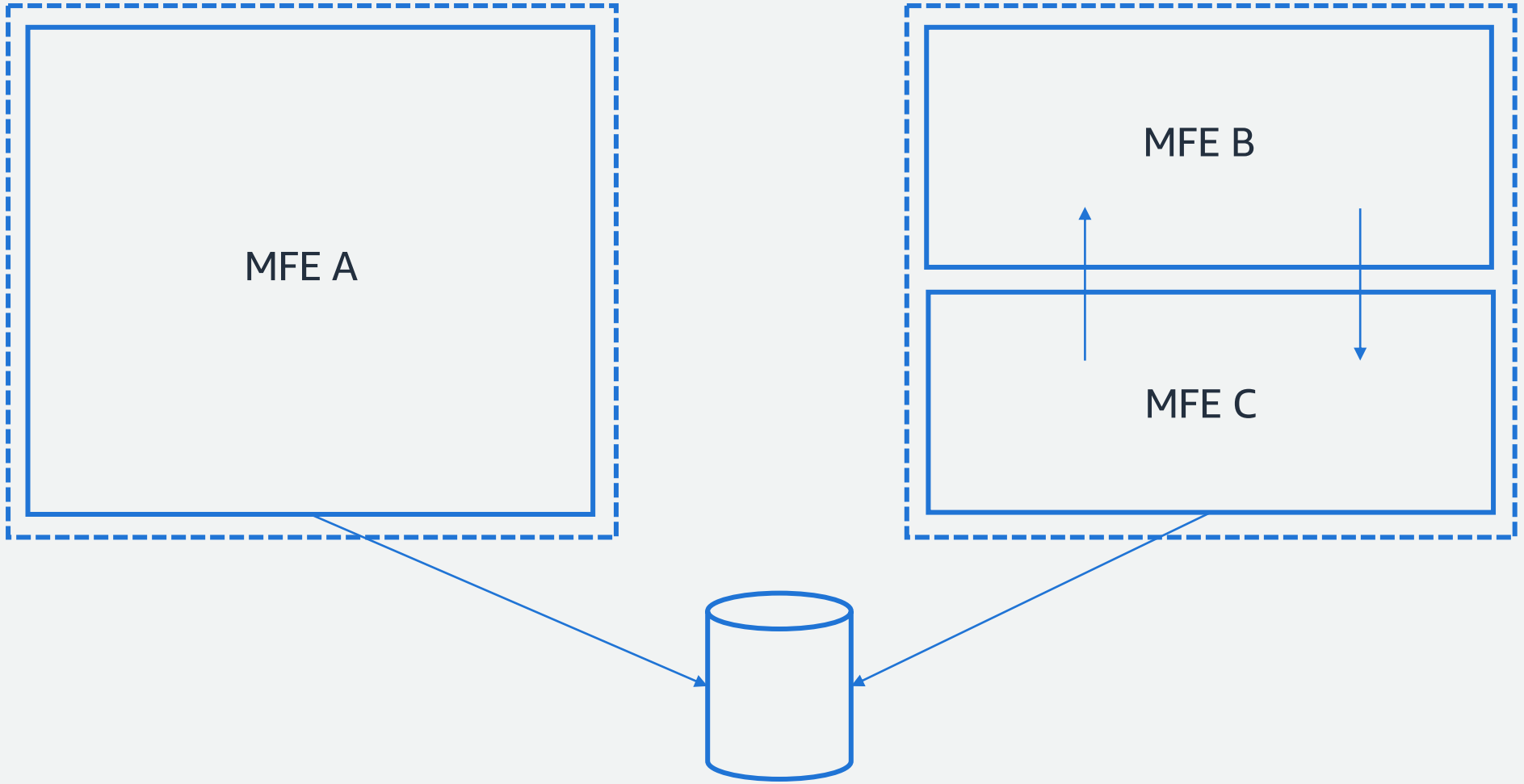
3 Após adquirir novas empresas

Use multi-frameworks quando apropriado, não otimize sua arquitetura para eles.

"The swiss army knife

(Escreva programas que fazem uma coisa e fazem bem)"

O projeto...



O editor legado

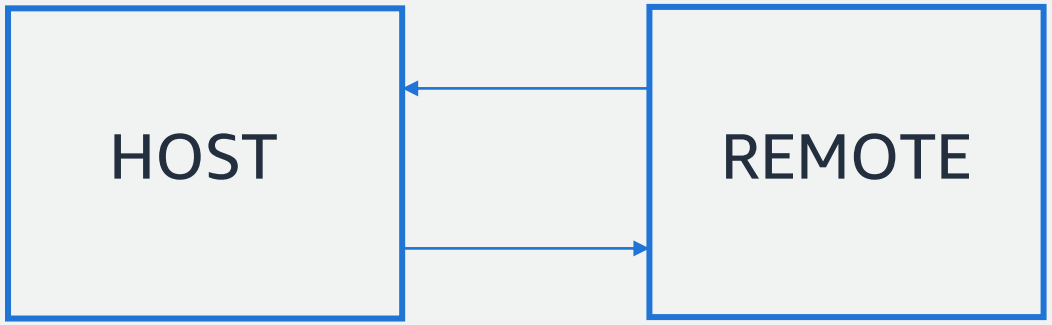
- Camada anticorrupção
- Alinhamento de decisões de arquitetura
- Prontos para o futuro

Poupe a base de código do shell, use uma camada anticorrupção para o sistema legado.

“A return ticket, please”

(Fluxo de dados unidirecional no resgate)

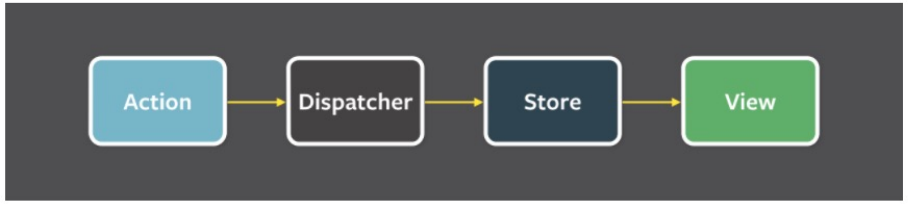
Compartilhamento



Fluxo de dados unidirecional no resgate

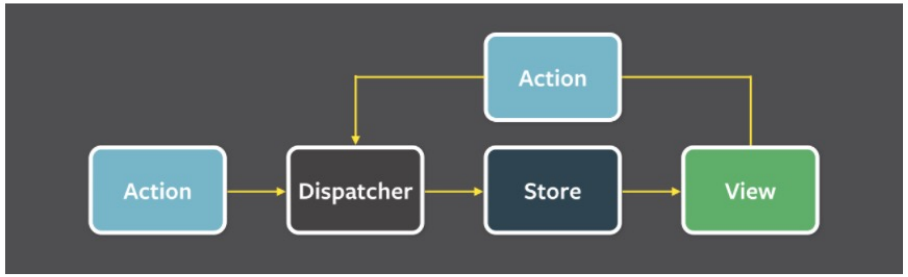
Structure and Data Flow

Data in a Flux application flows in a single direction:



A unidirectional data flow is central to the Flux pattern, and the above diagram should be **the primary mental model for the Flux programmer**. The dispatcher, stores and views are independent nodes with distinct inputs and outputs. The actions are simple objects containing the new data and an identifying *type* property.

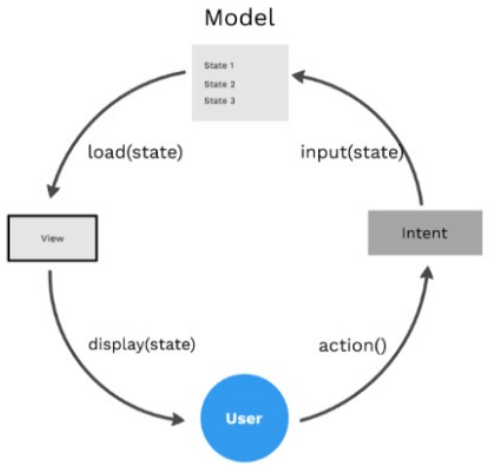
The views may cause a new action to be propagated through the system in response to user interactions:



All data flows through the dispatcher as a central hub. Actions are provided to the dispatcher in an *action creator* method, and most often originate from user interactions with the views. The dispatcher then invokes the callbacks that the stores have registered with it, dispatching actions to all stores. Within their registered callbacks, stores respond to whichever actions are relevant to the state they maintain. The stores then emit a *change* event to alert the controller-views that a change to the data layer has occurred. Controller-views listen for these events and retrieve data from the stores in an event handler. The controller-views call their own `setState()` method, causing a re-rendering of themselves and all of their descendants in the component tree.

How does the MVI work?

User does an action which will be an Intent → Intent is a state which is an input to model → Model stores state and send the requested state to the View → View Loads the state from Model → Displays to the user. If we observe, the data will always flow from the user and end with the user through intent. It cannot be the other way, Hence its called Unidirectional architecture. If the user does one more action the same cycle is repeated, hence it is Cyclic.



MVI cyclic flow representation

Aprendizados de fluxo de dados unidirecionais

1 Fácil de depurar

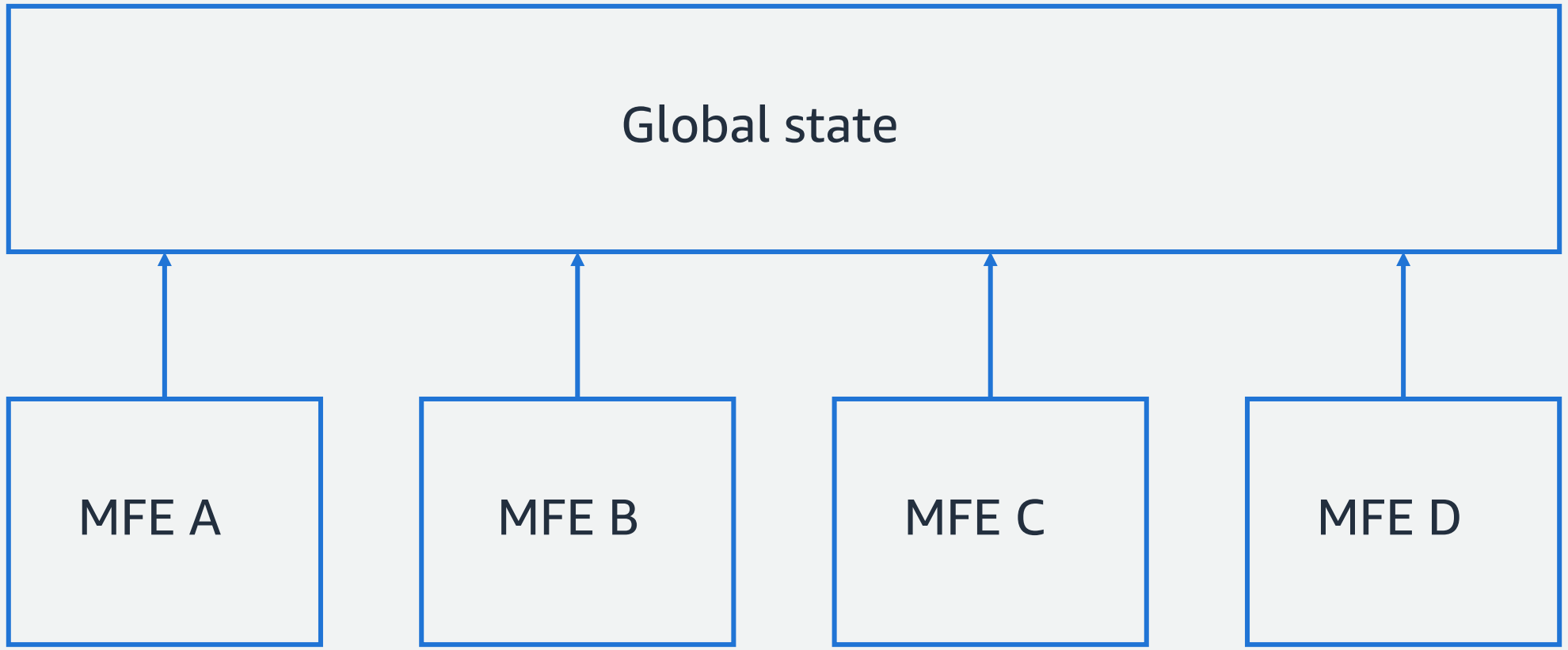
2 Menos propenso a erros

Evite o compartilhamento bidirecional a menos que estritamente necessário

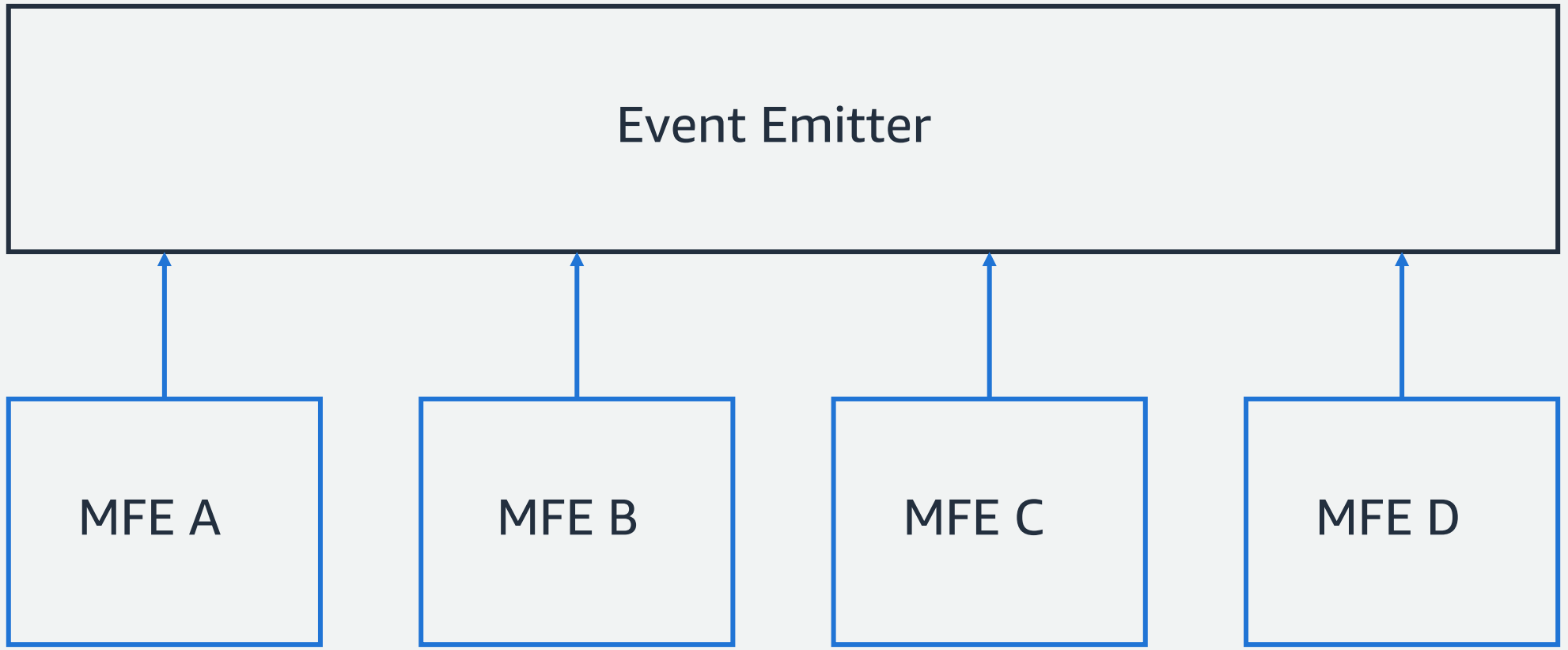
“Relax, it’s just code”

(Evite o acoplamento organizacional)

Acoplamento em tempo de projeto



Entidades fracamente acopladas

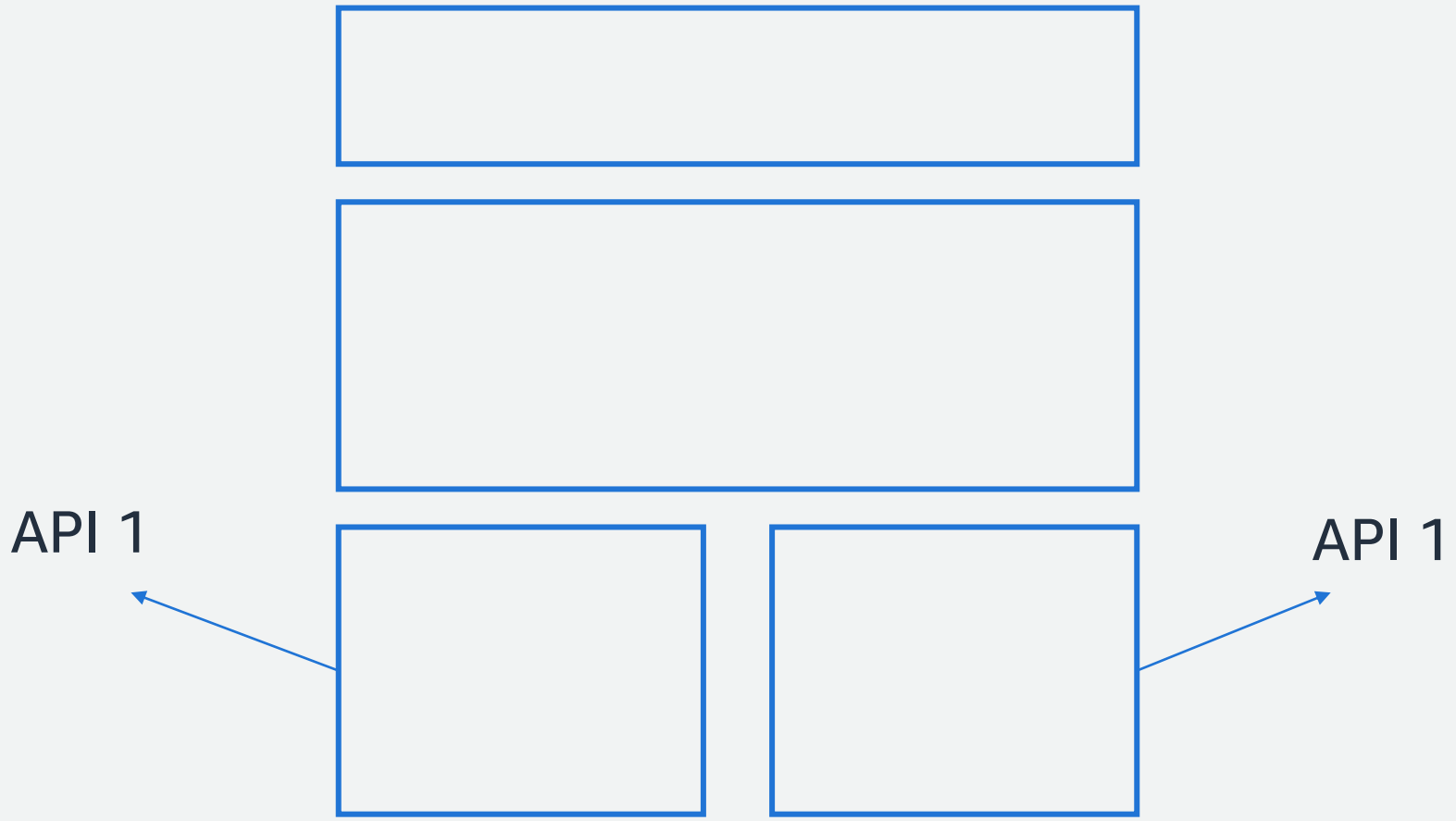


Adote MFEs pouco acoplados, mas altamente alinhados

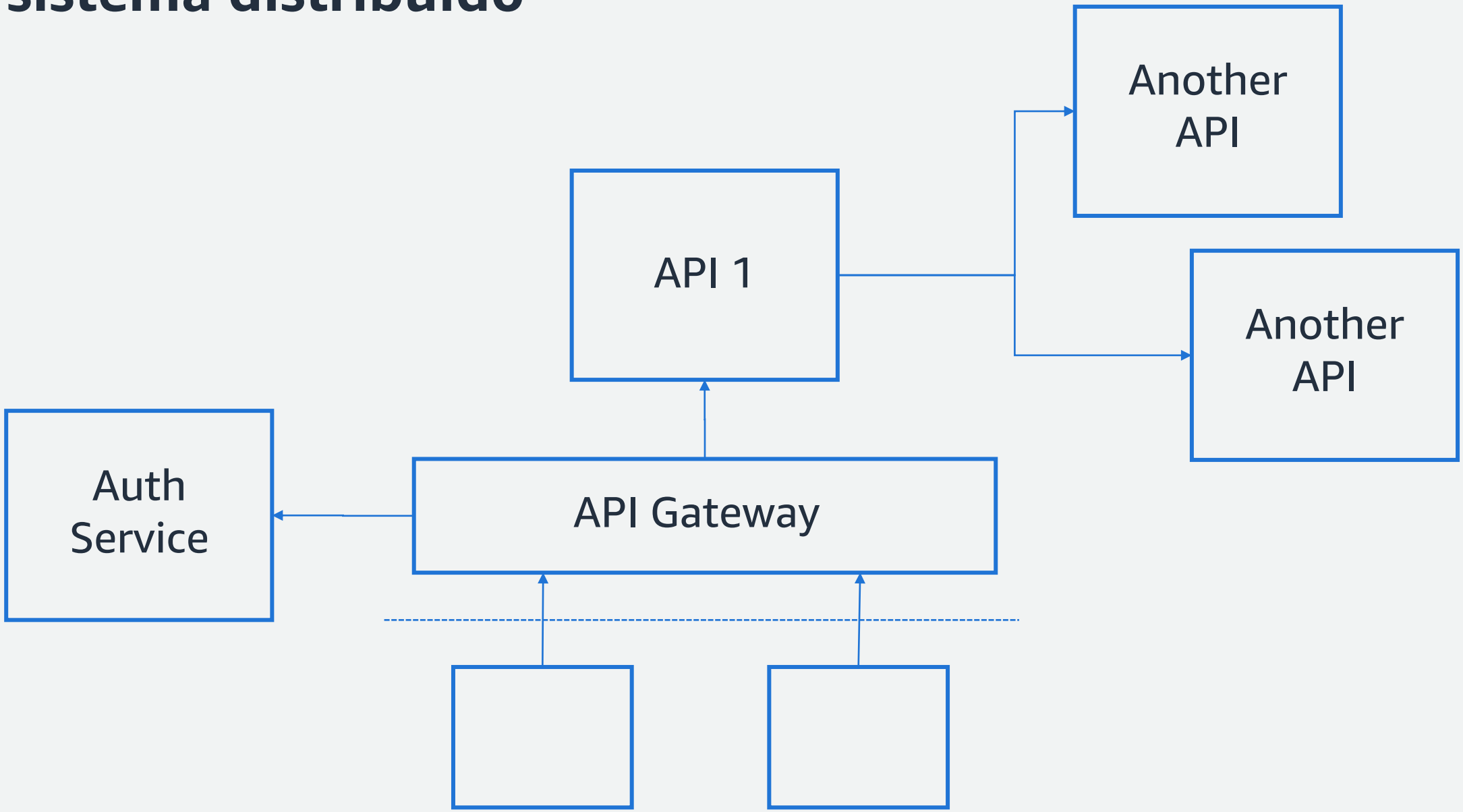
“Let’s hammer the APIs”

(Várias MFEs chamando o mesmo ponto de extremidade)

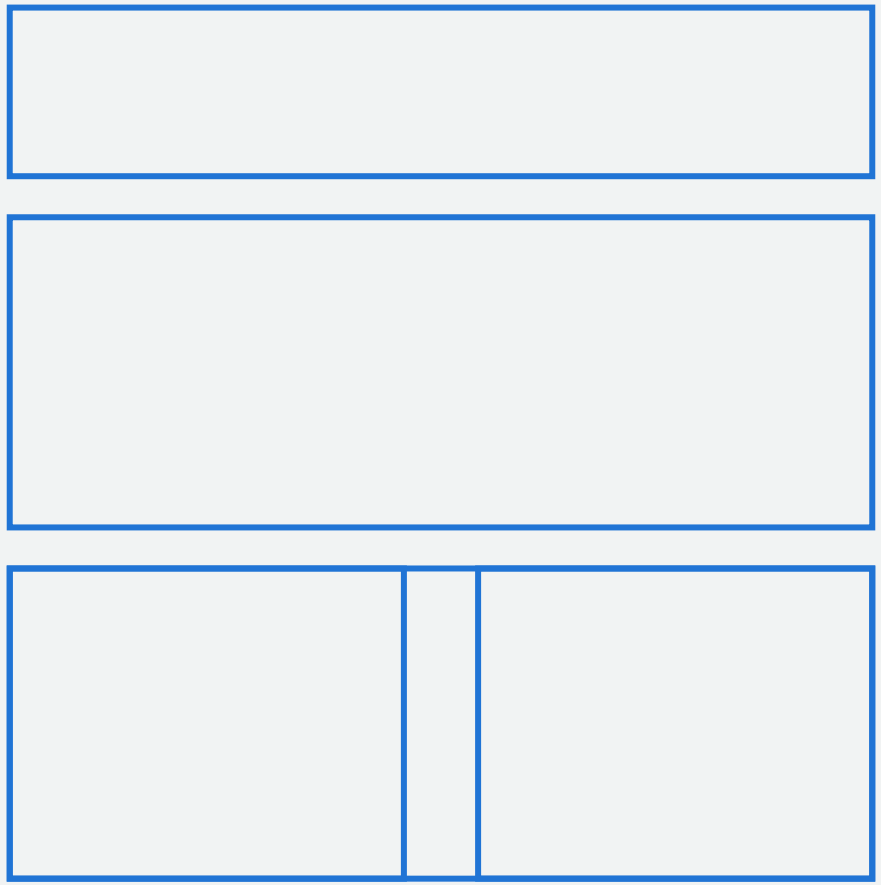
Vários MFEs na mesma view



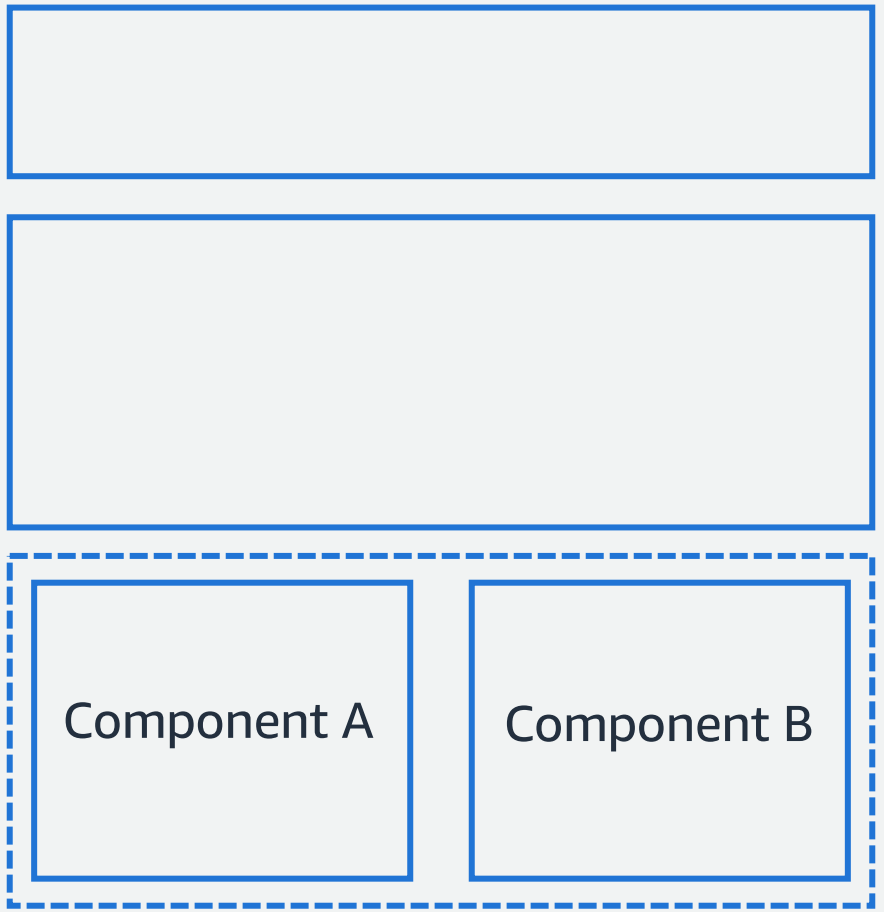
Seu sistema distribuído



Soluções possíveis



Soluções possíveis

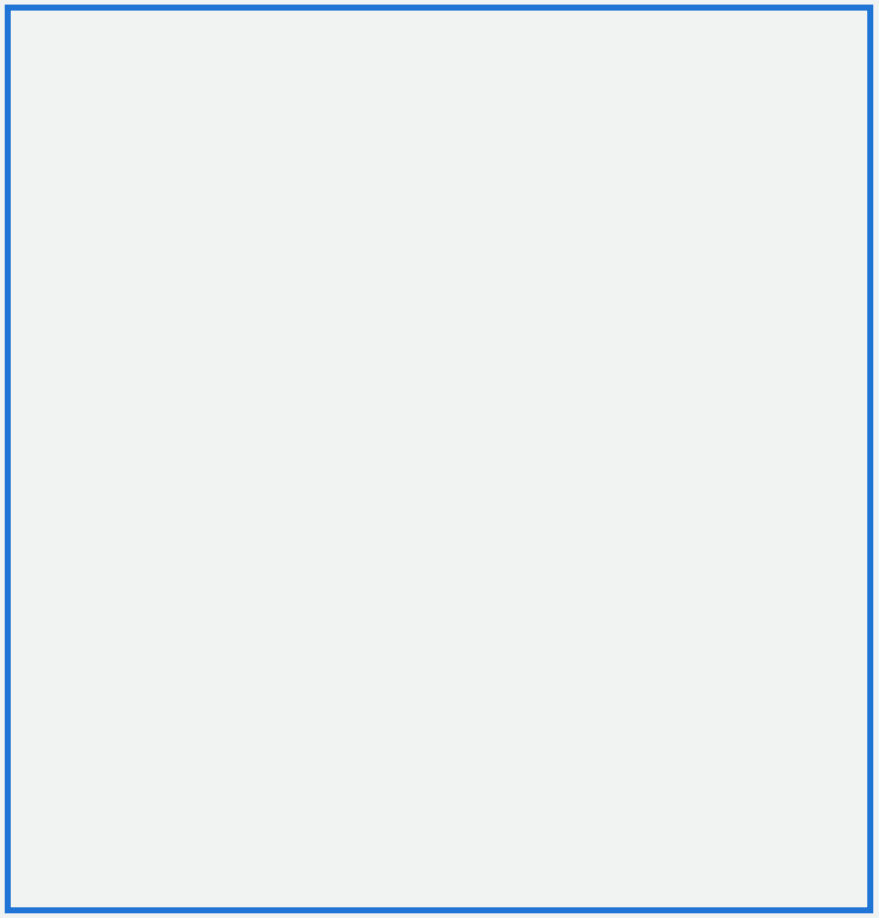


Entenda o impacto de ponta a ponta das suas decisões

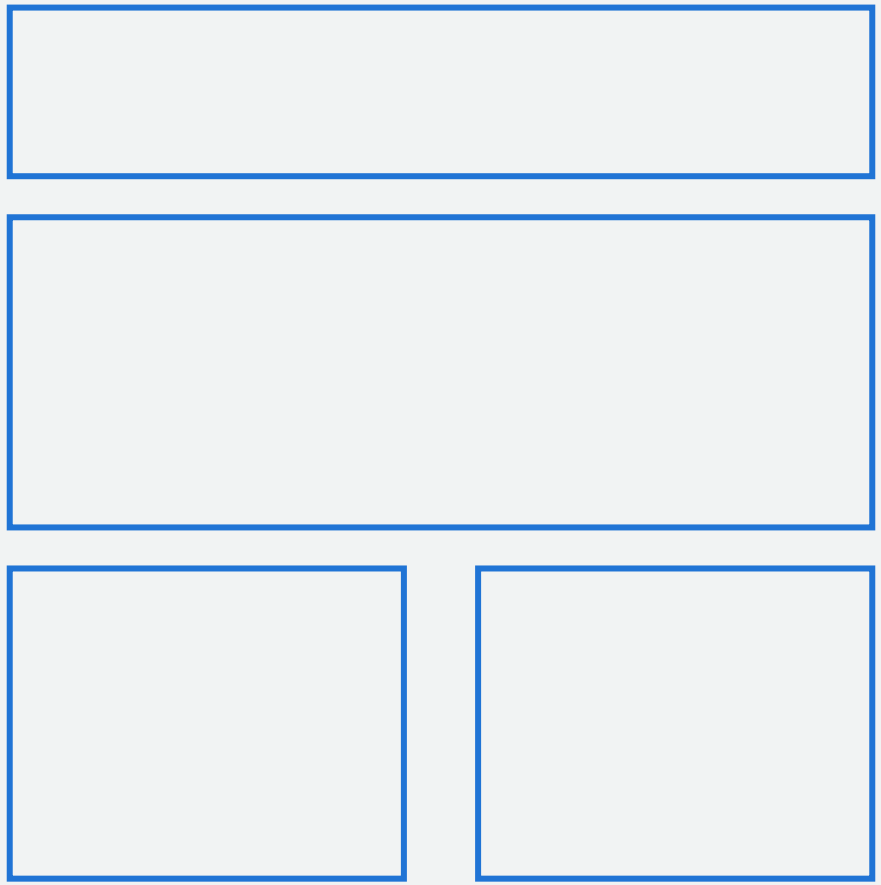
"Bye bye big-bang"

(Implantação iterativa)

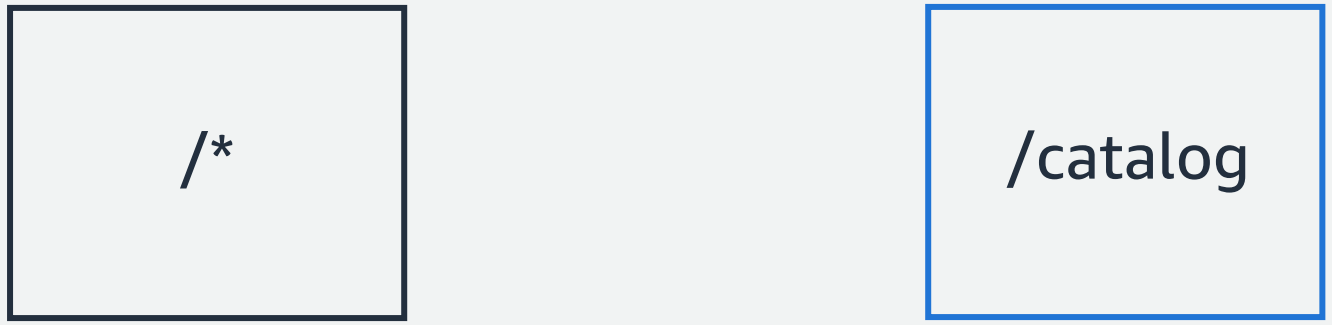
Estratégias de migração



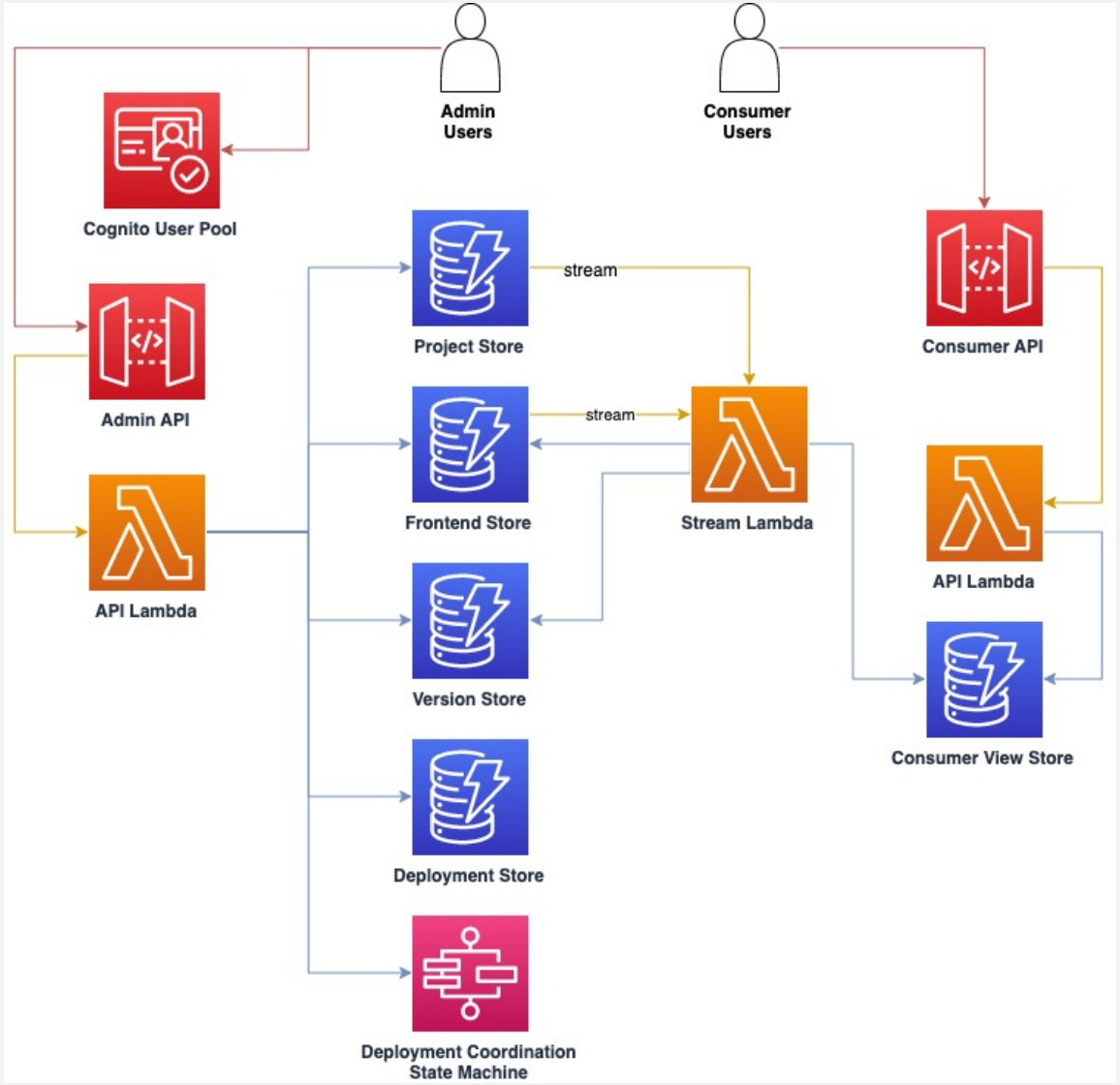
Estratégias de migração



Estratégias de migração



Estratégias de migração



https://bit.ly/3G0xnNR

A implantação iterativa de Micro-frontends ajuda aumentando a confiança dos desenvolvedores além de agregar valor para os usuários.

A arquitetura é sempre um trade-off, *basta encontrar uma abordagem equilibrada para o seu contexto.*

Obrigado!

Pedro Henrique Oliveira
pedrohco@

Ricardo Tasso
riribeir@

Survey: