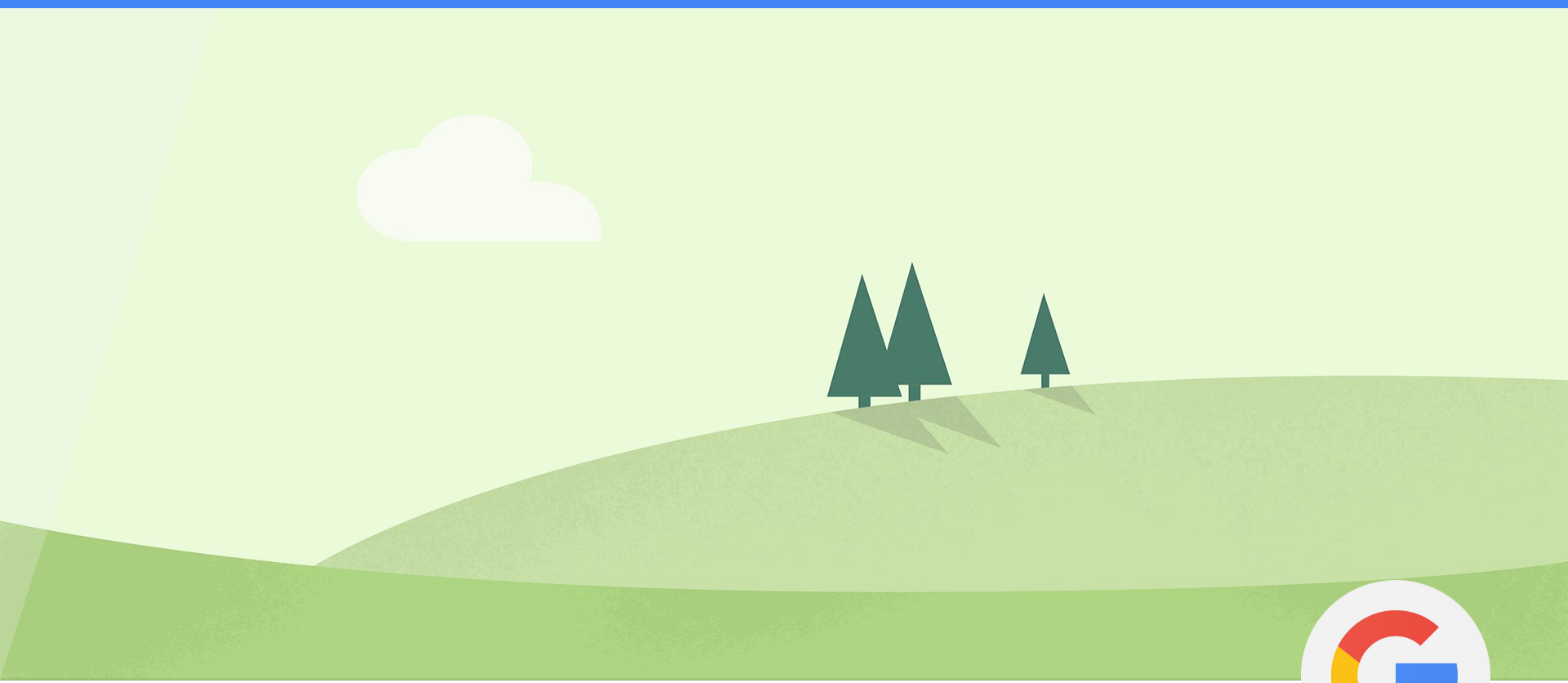


A vida além do event sourcing

legal e perigoso ao mesmo tempo

Dr. Biharck Araújo, Principal Architect, Google (linkedin.com/in/biharck)

Biharck Araújo

Principal Architect

Google

"We can query an application's state to find out the current state of the world, and this answers many questions. However there are times when we don't just want to see where we are, we also want to know how we got there."

- Martin Fowler

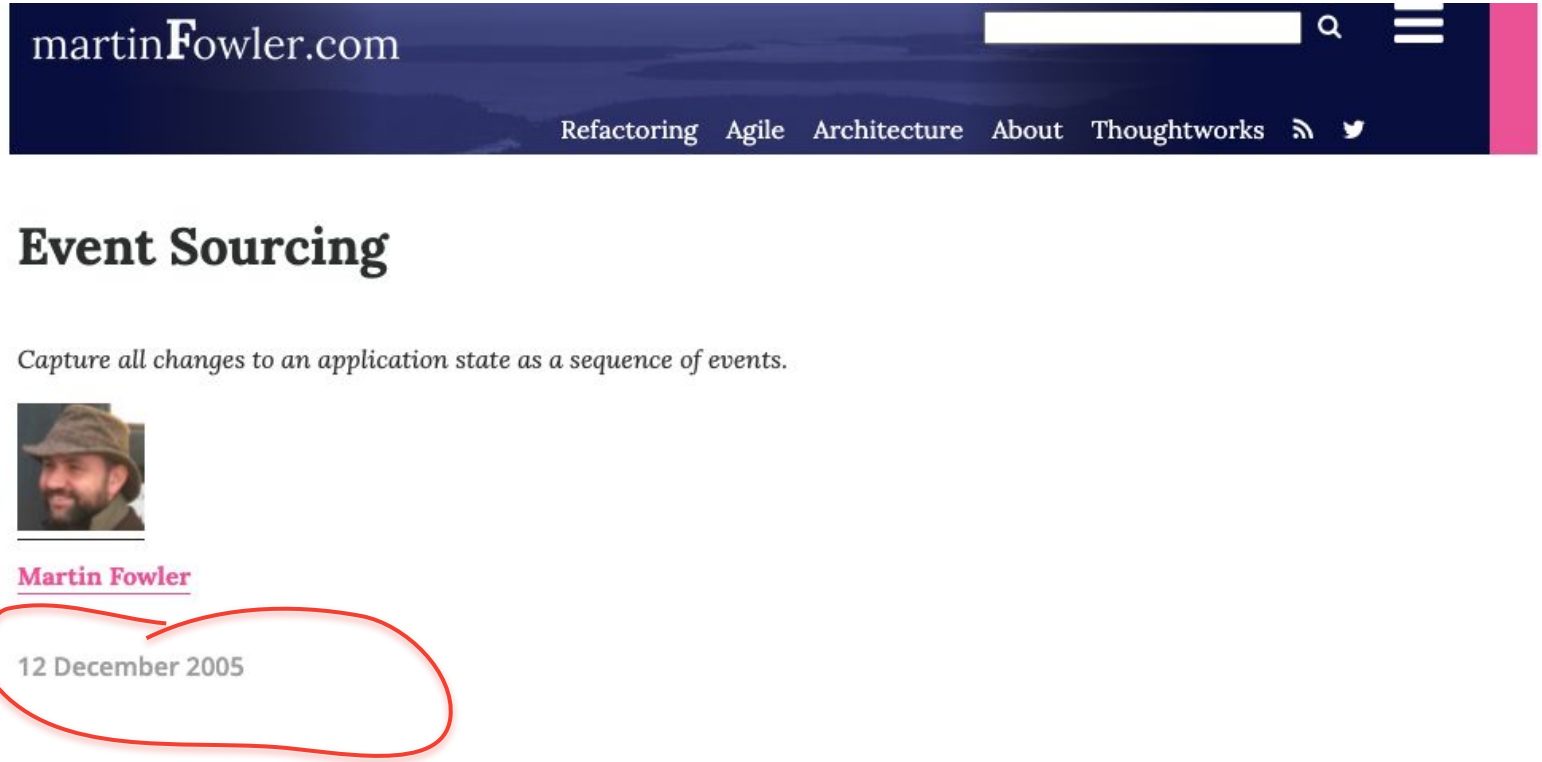
Event Sourcing

O que vamos falar hoje 

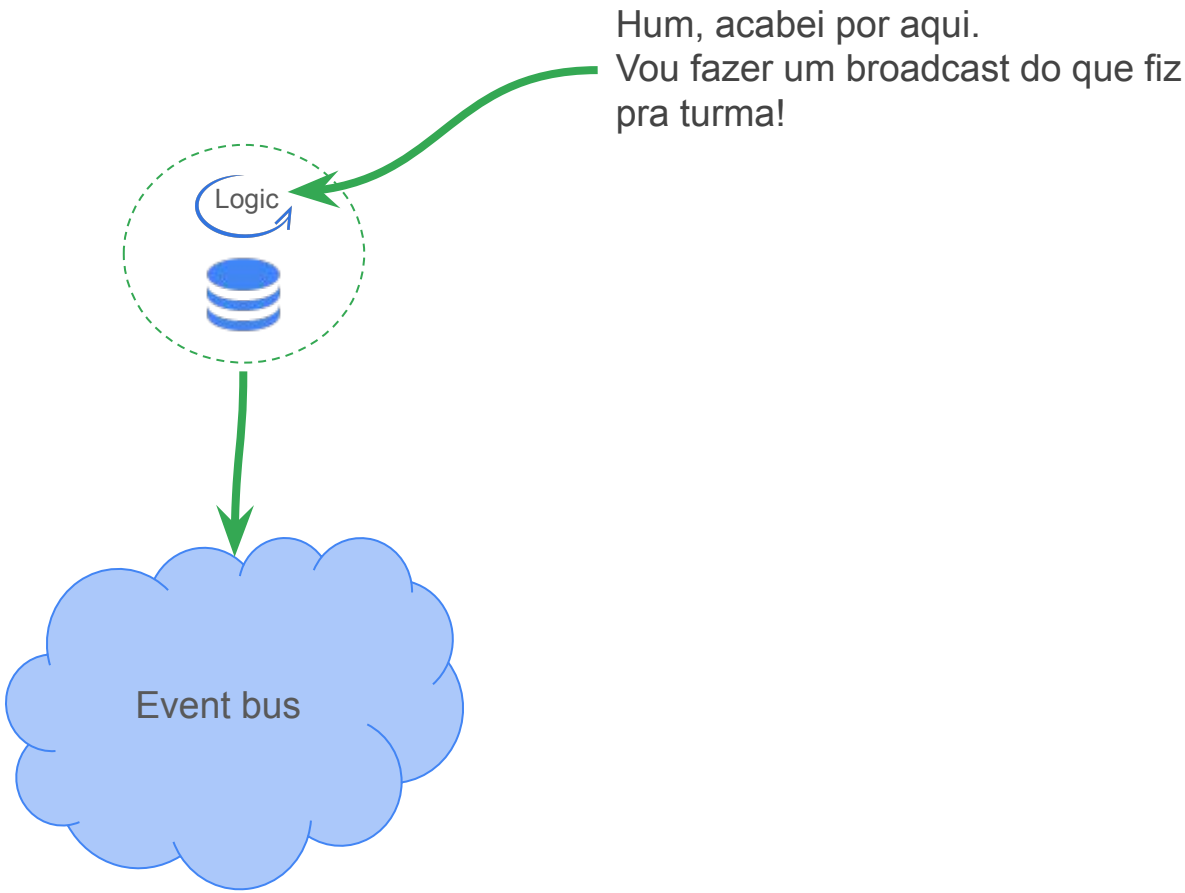
- As partes mais importantes de uma estratégia de Event Sourcing
- Os erros mais comuns em projetos baseados em Event Sourcing
    - Como contornar e resolver

Back to basics - Event-Driven

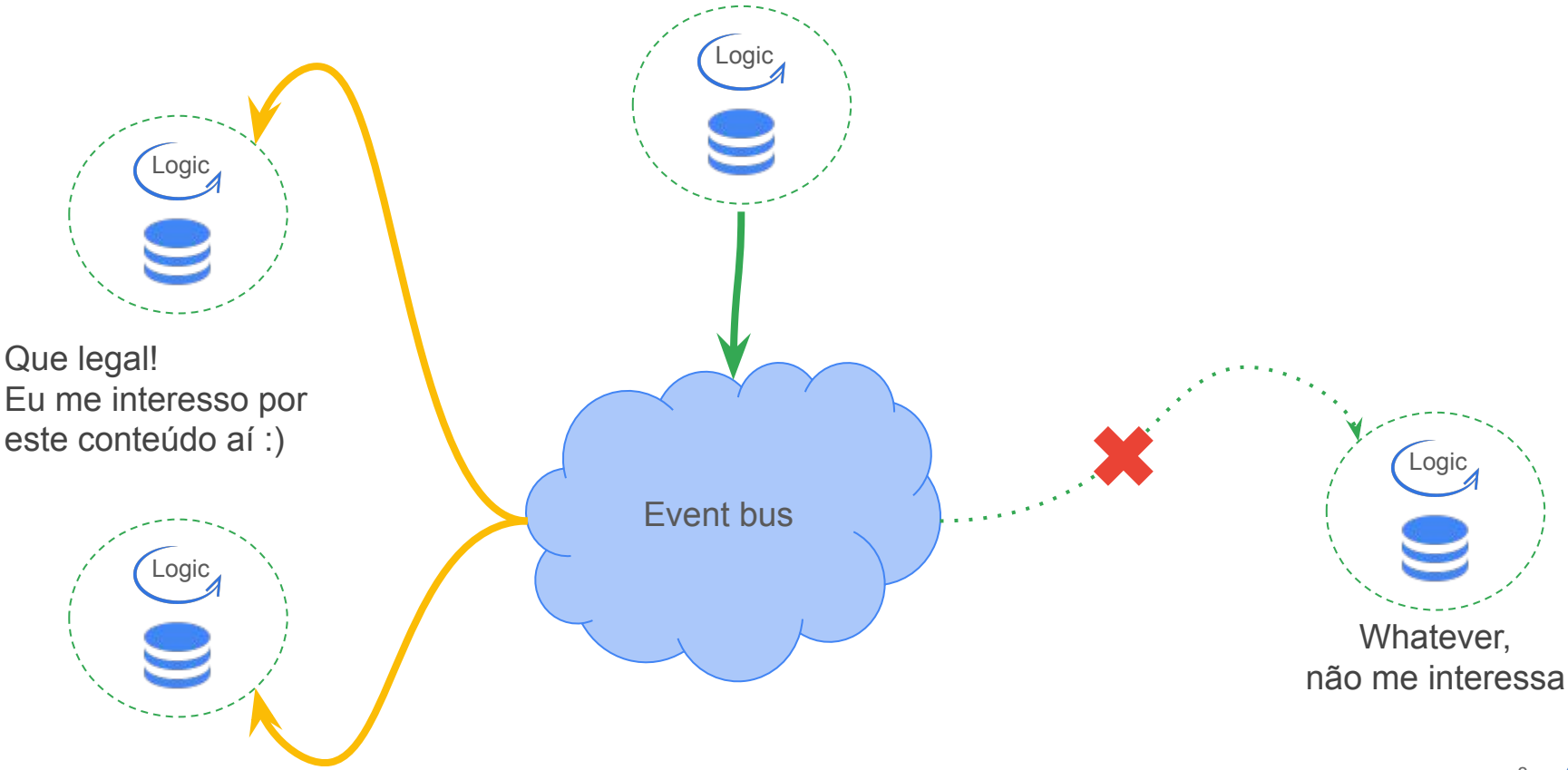
Spoiler Alert



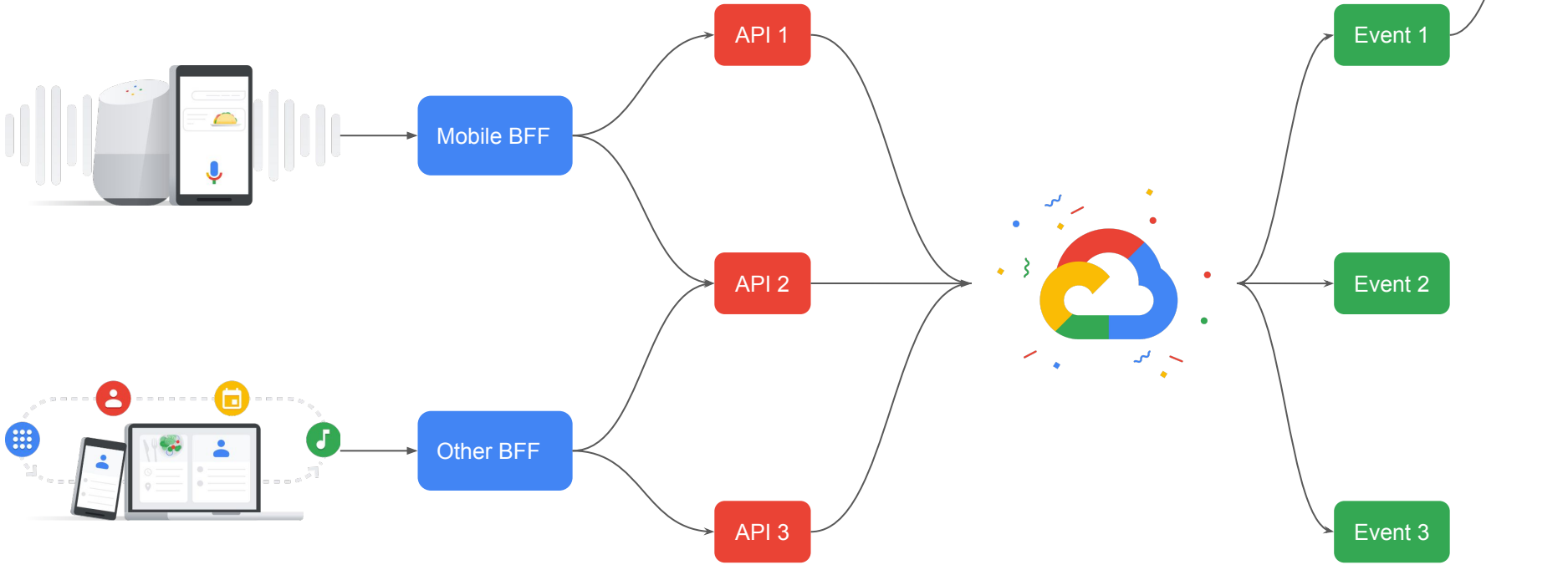
Event-Driven



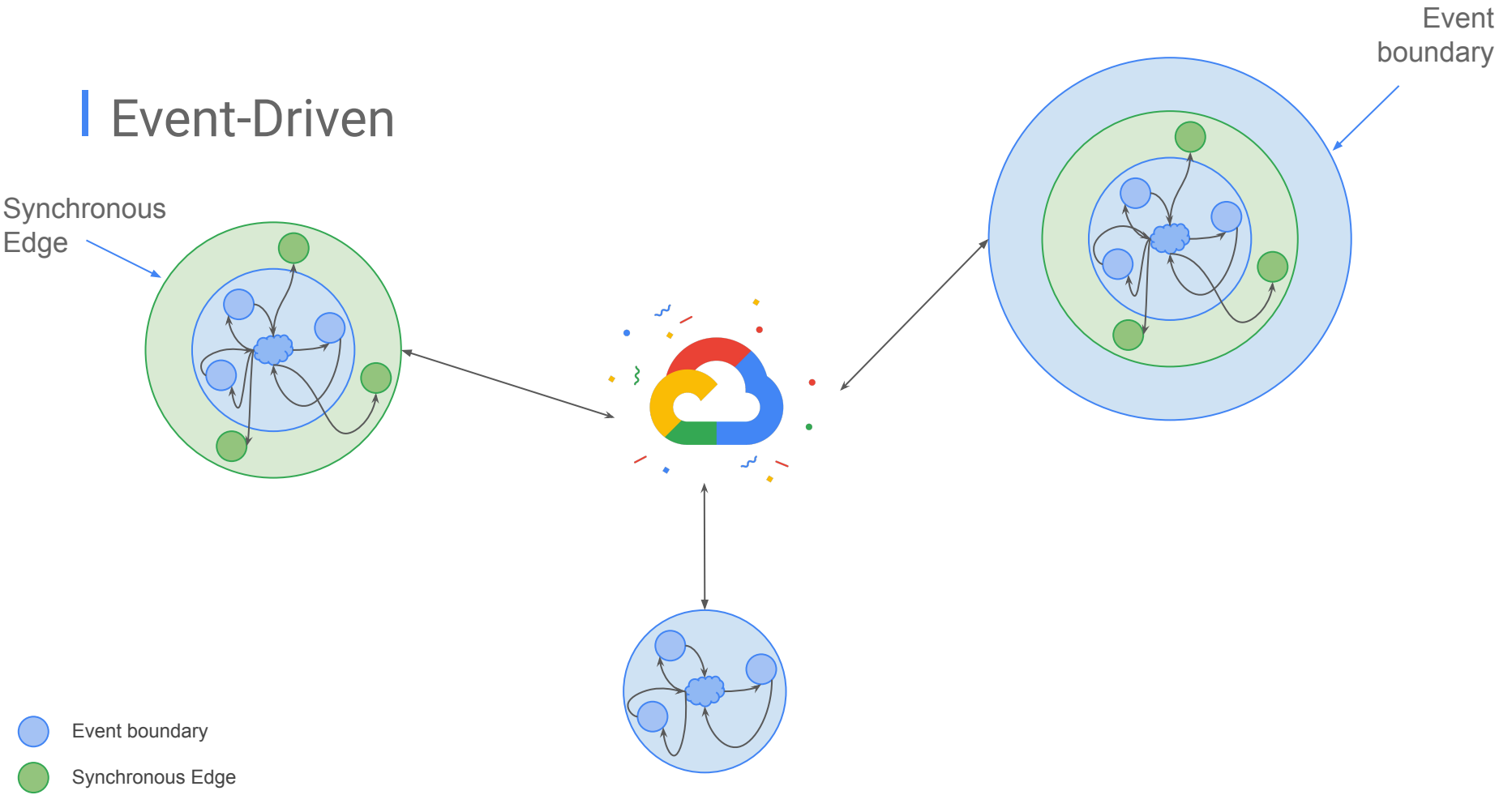
Event-Driven



Event-Driven

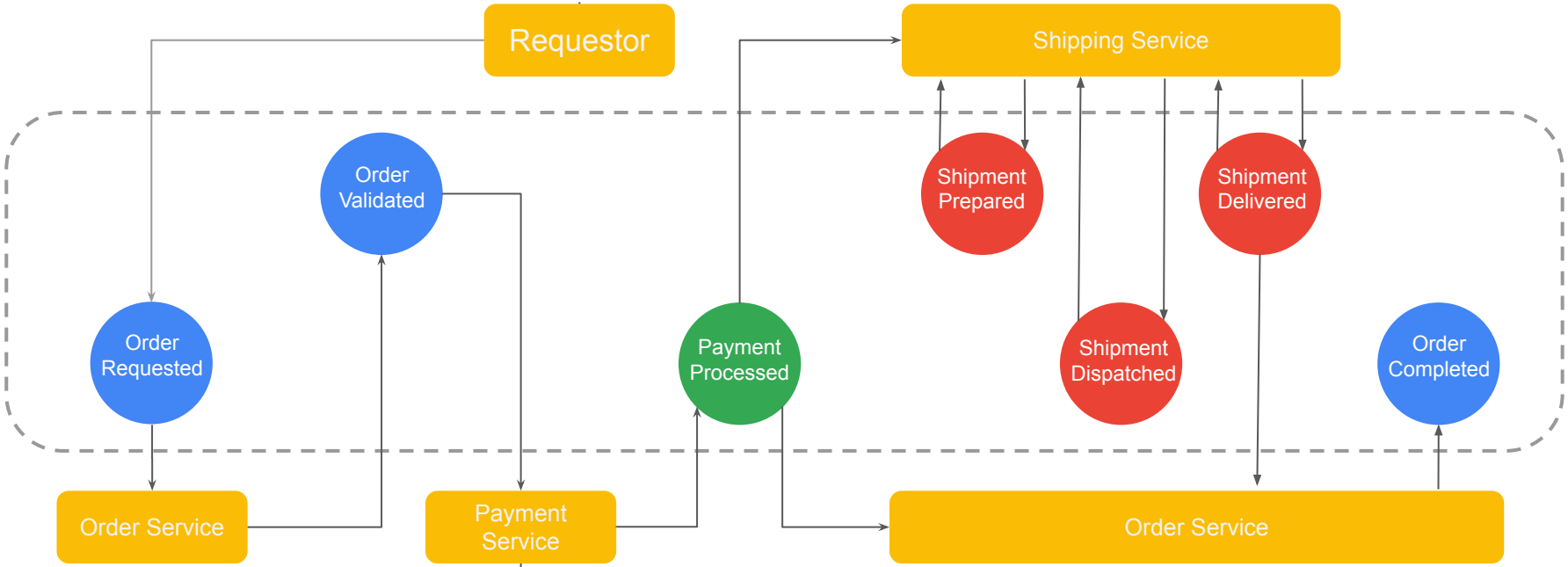


Event-Driven



Fonte: https://ookami86.github.io/event-sourcing-in-practice/

Event-Driven

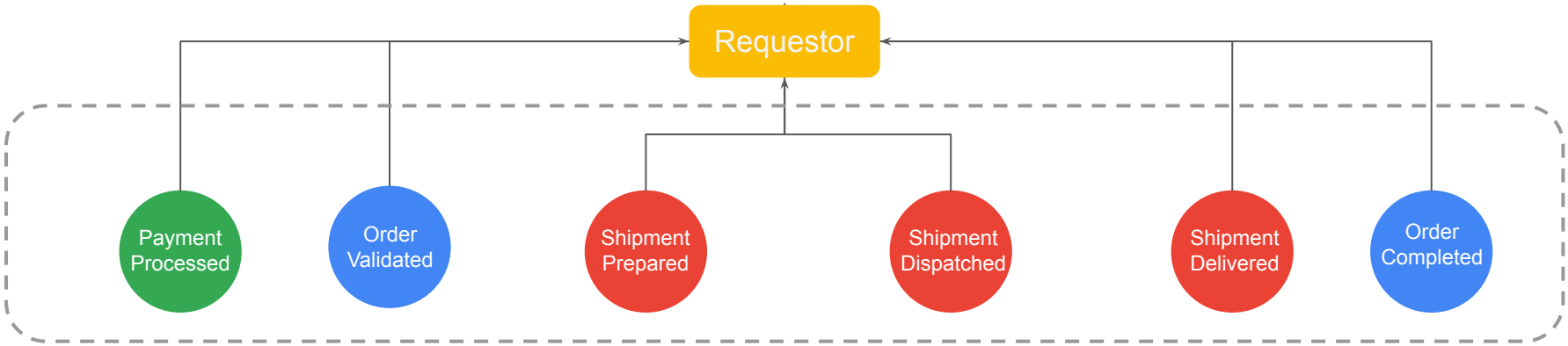


Shipments Topic

Payments Topic

Orders Topic

Event-Driven



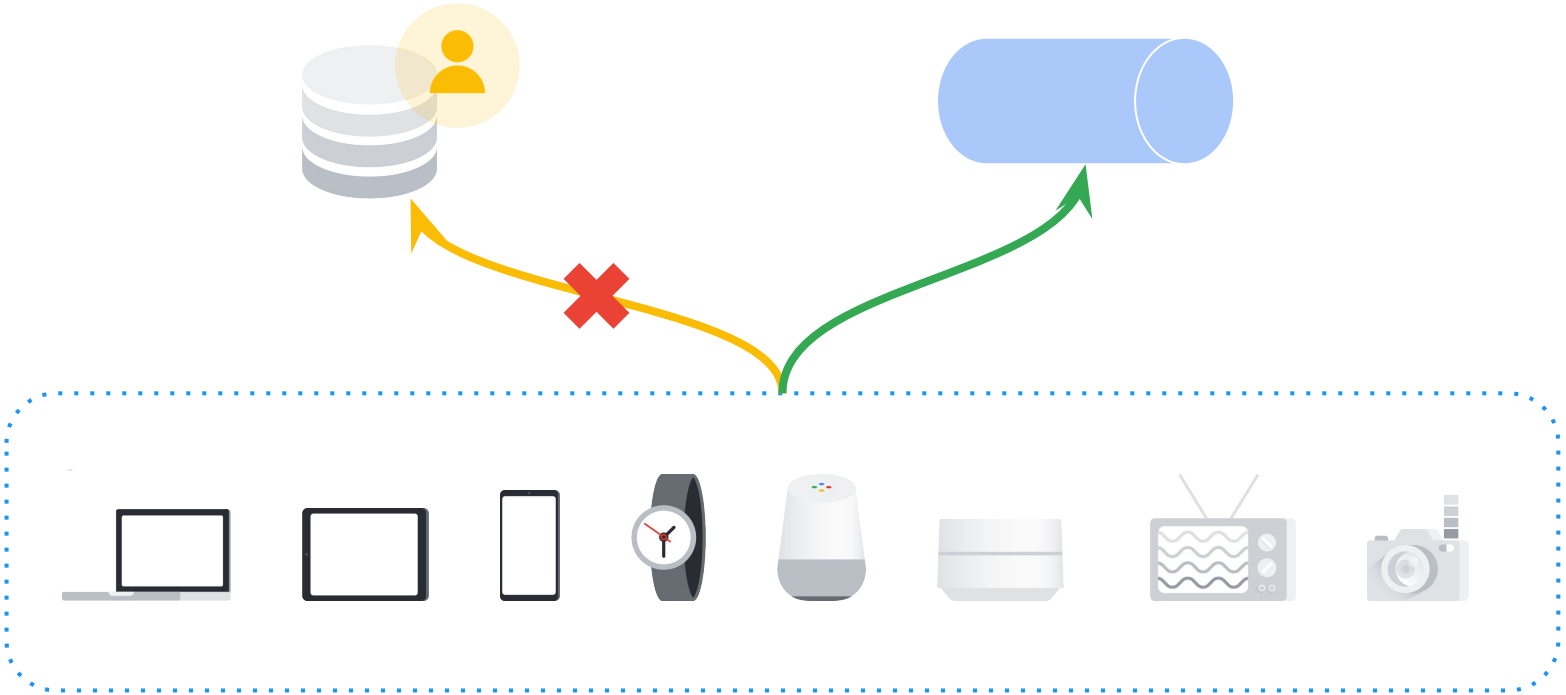
Shipments Topic

Payments Topic

Orders Topic

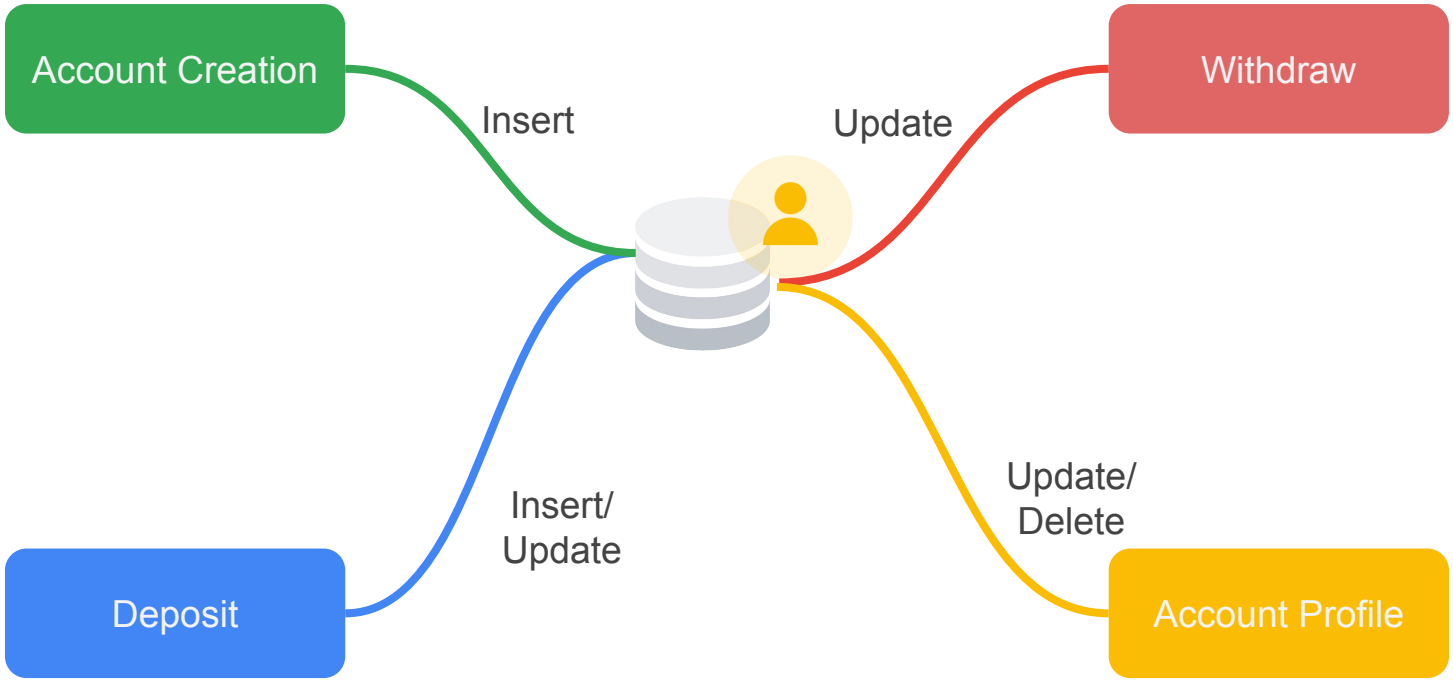
Event-Sourcing

Event Sourcing



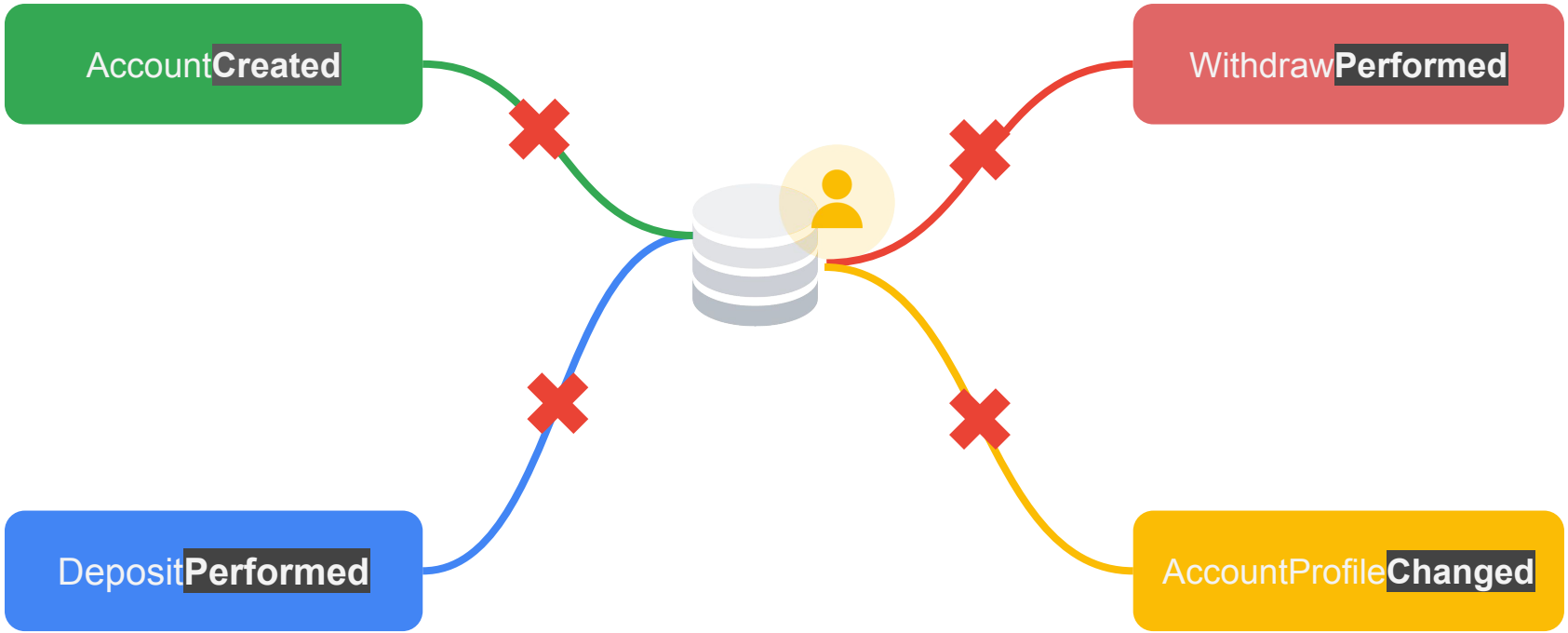
Event Sourcing

Como isso funciona?



Event Sourcing

Como isso funciona?



Event Sourcing

Como isso funciona?

AccountCreated

AccountId: *102030*
Owner: *Biharck*
Country: *Brazil*

WithdrawPerformed

AccountId: *102030*
Amount: *$20*

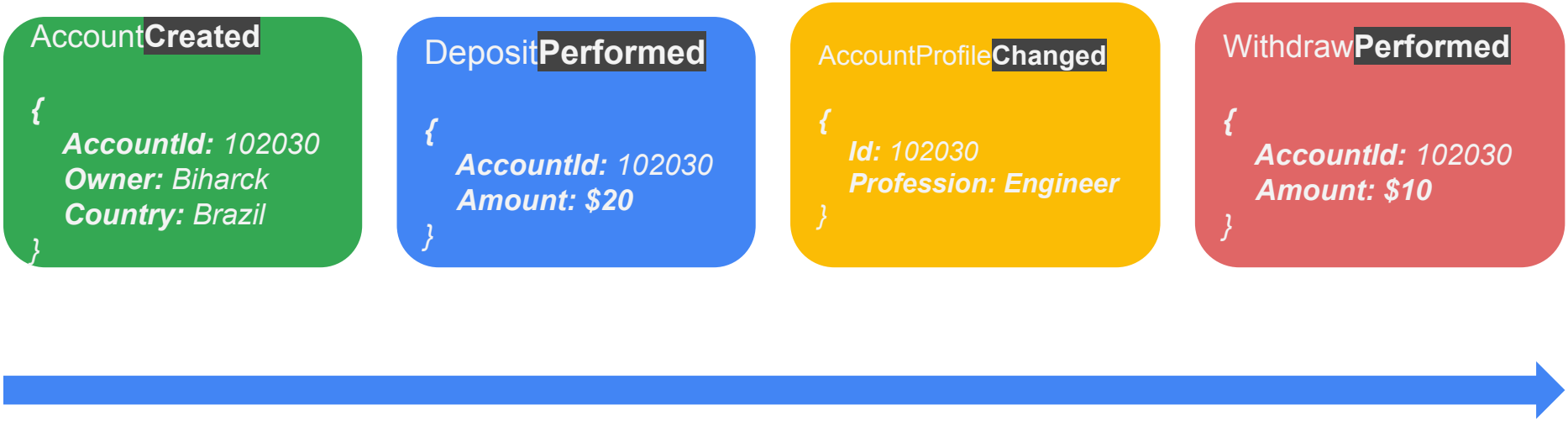
DepositPerformed

AccountId: *102030*
Amount: *$10*

AccountProfileChanged

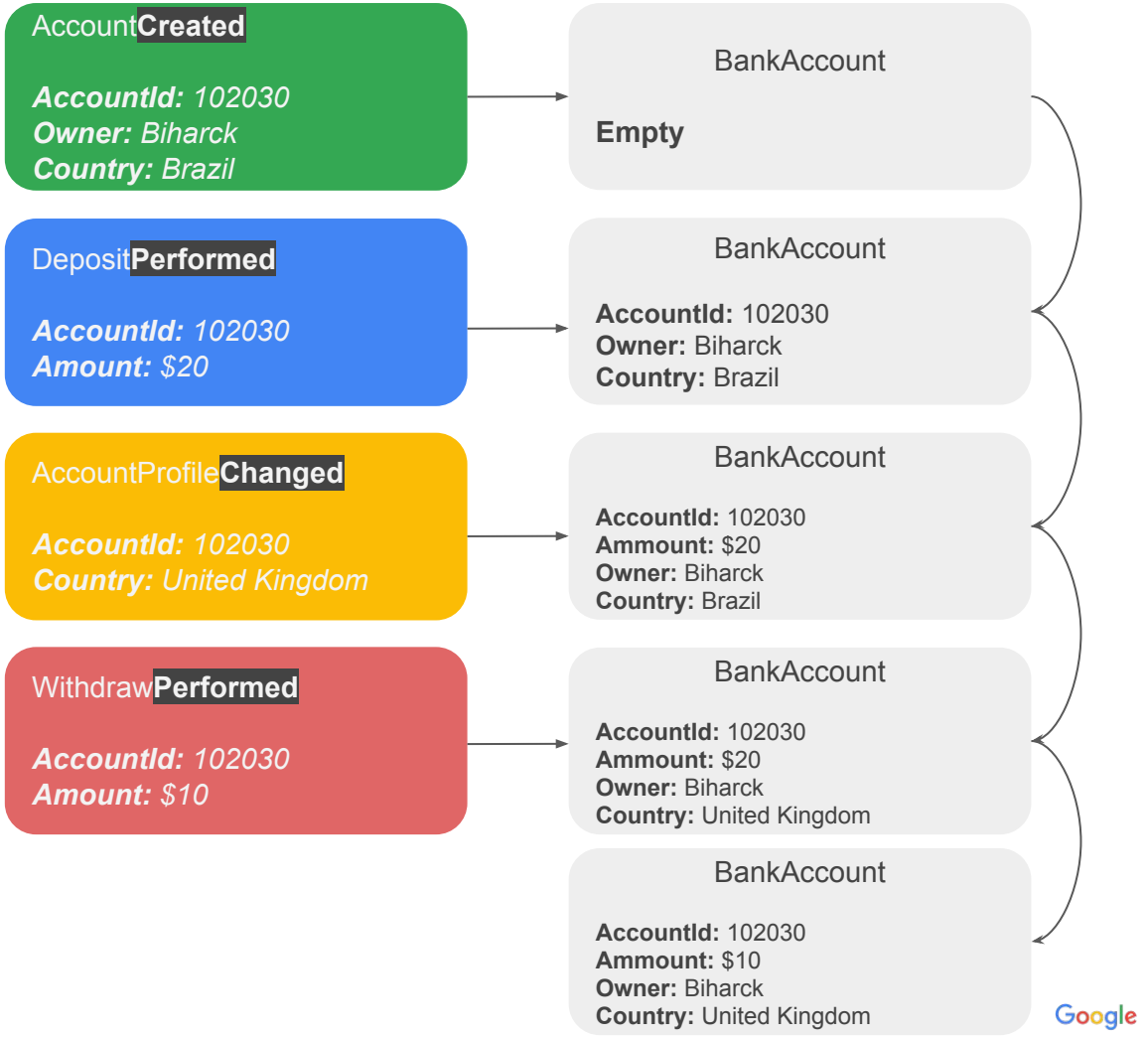
AccountId: *102030*
Profession: *Engineer*

Event Sourcing



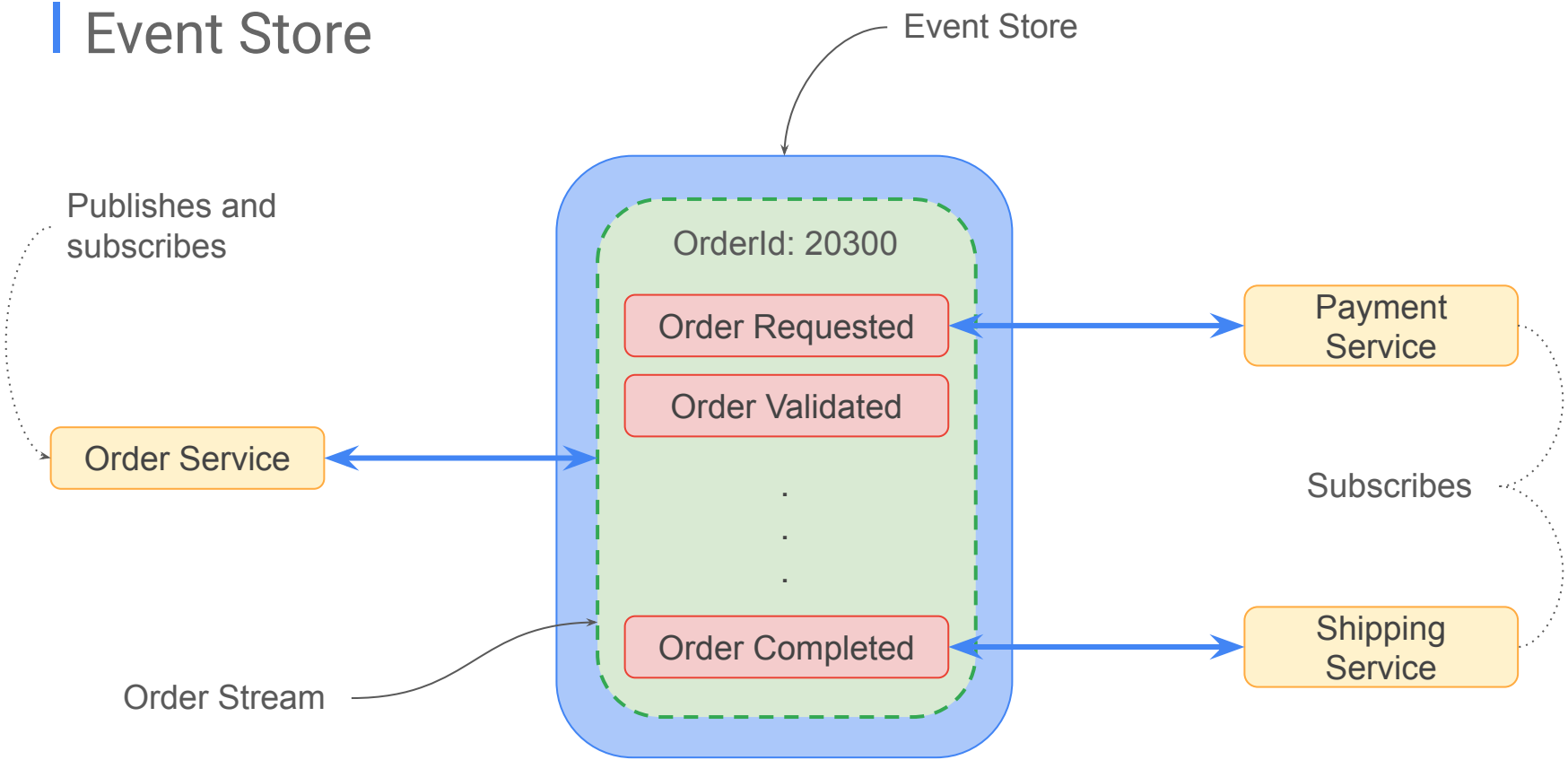
Event Sourcing

Restoring



Event Store

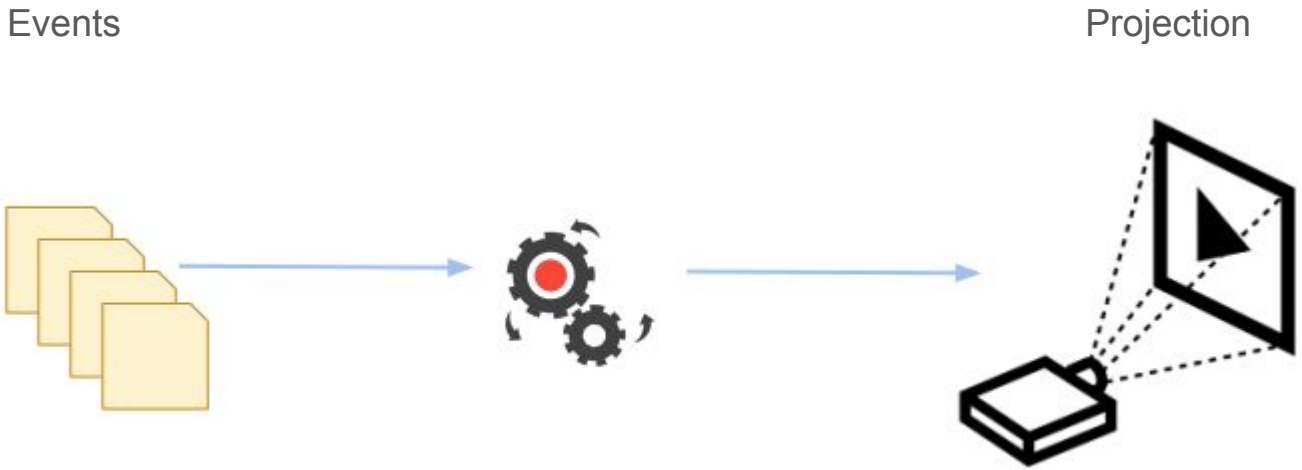
Event Store


Event Store
Publishes and subscribes
OrderId: 20300
Order Requested
Order Validated
Order Completed
Payment Service
Order Service
Subscribes
Shipping Service
Order Stream

E por que é tão legal?

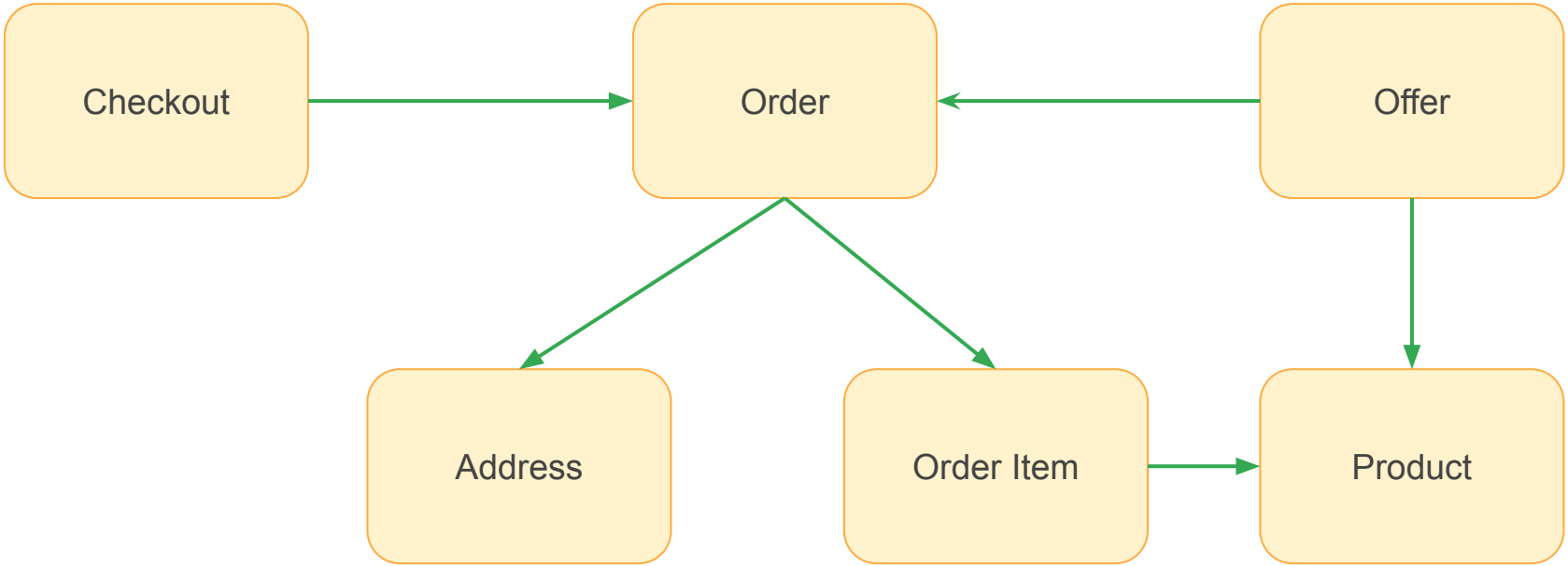
Event Sourcing

Ephemeral Data-structures



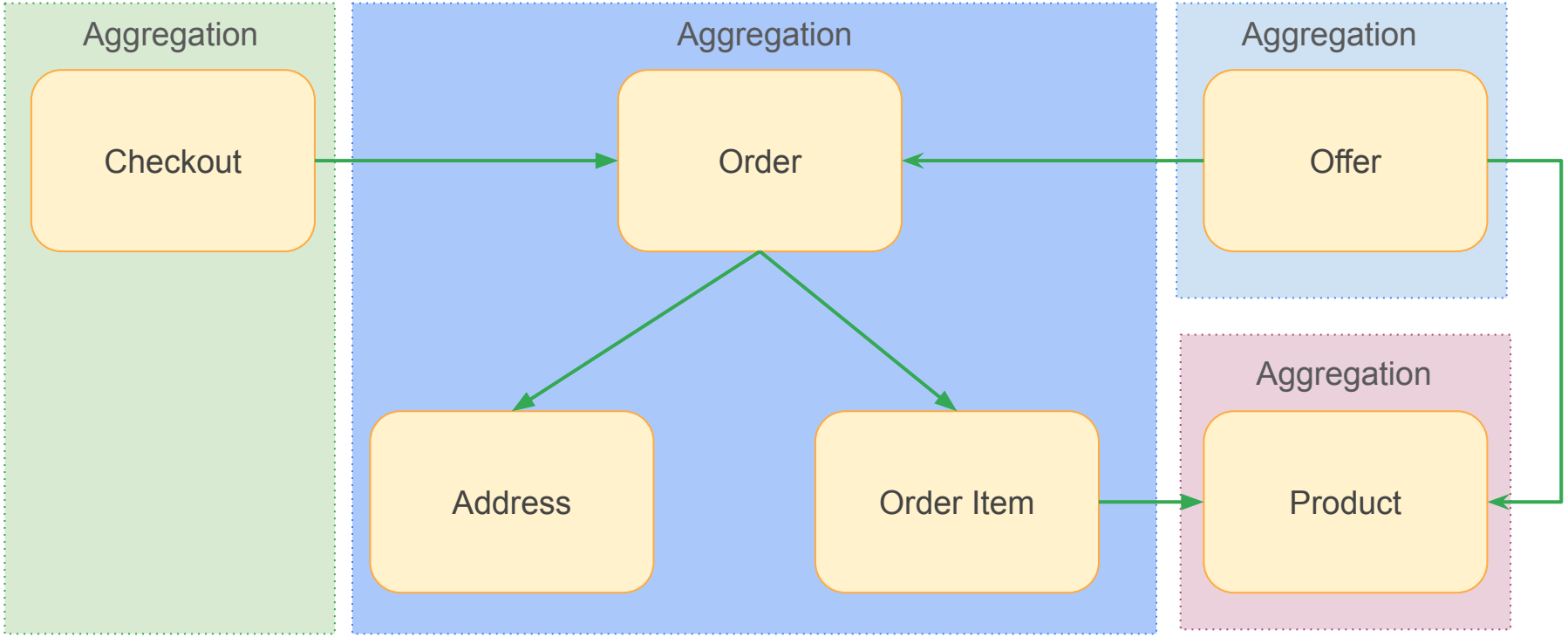
Event Sourcing

Easier to communicate with domain experts



Event Sourcing

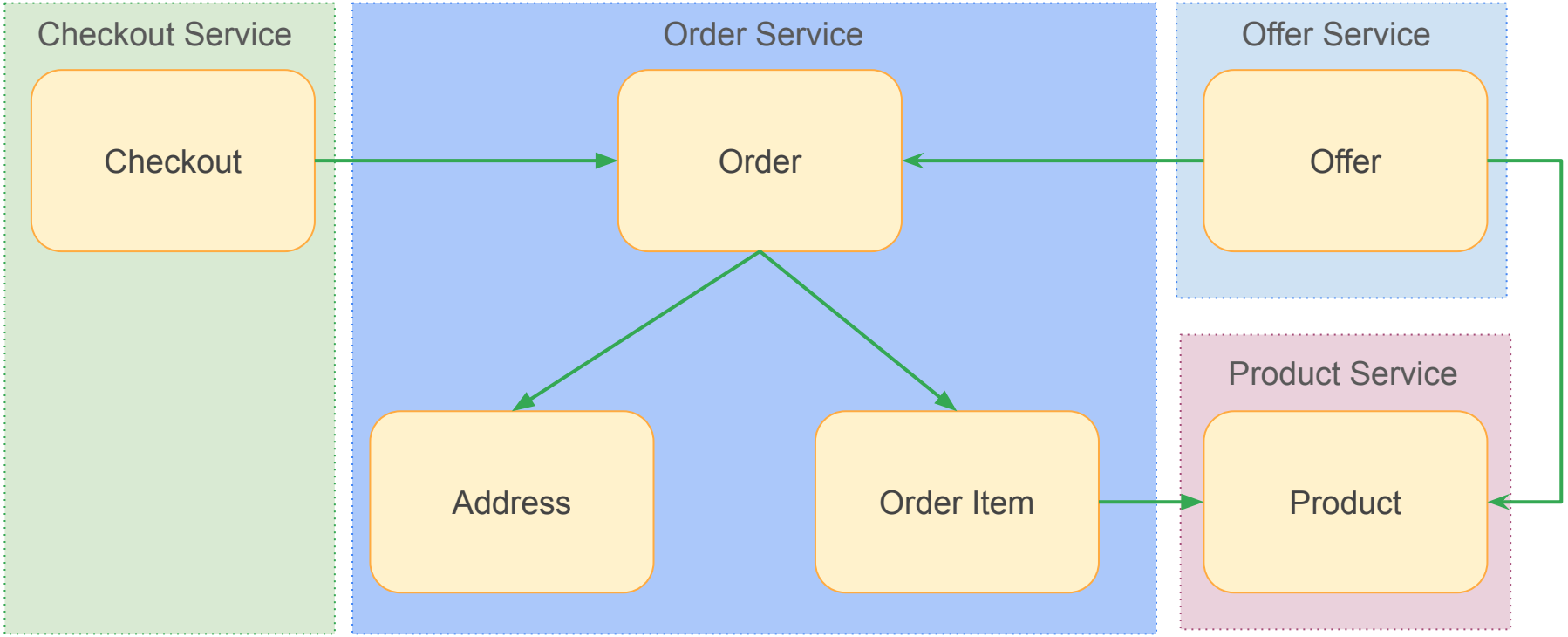
Domain Model = Loosely coupled aggregations



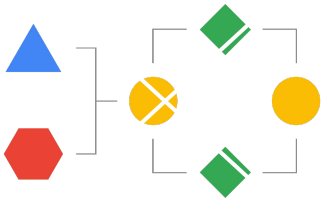
* Chris Richardson

Event Sourcing

Easier moving into microservices



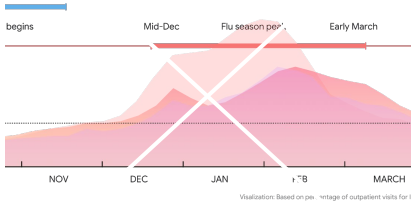
Event Sourcing



Expressive Models

Eventos são first class objects ao invés de implícito state change

Os modelos serão muito semelhantes aos processos reais que você está modelando



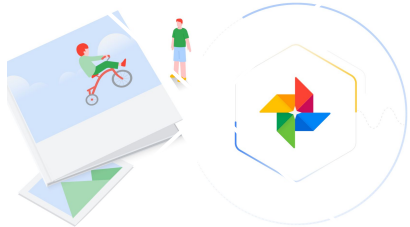
Painless access to Historical Data

Histórico completo de todos os eventos que já aconteceram
Todos os dados em uma ordem cronológica

Troubleshooting

A vida é curta demais para perder tempo com troubleshooting

Event Sourcing



Composing services is now actionable

Deixe os serviços se comunicarem por eventos

Legacy migration

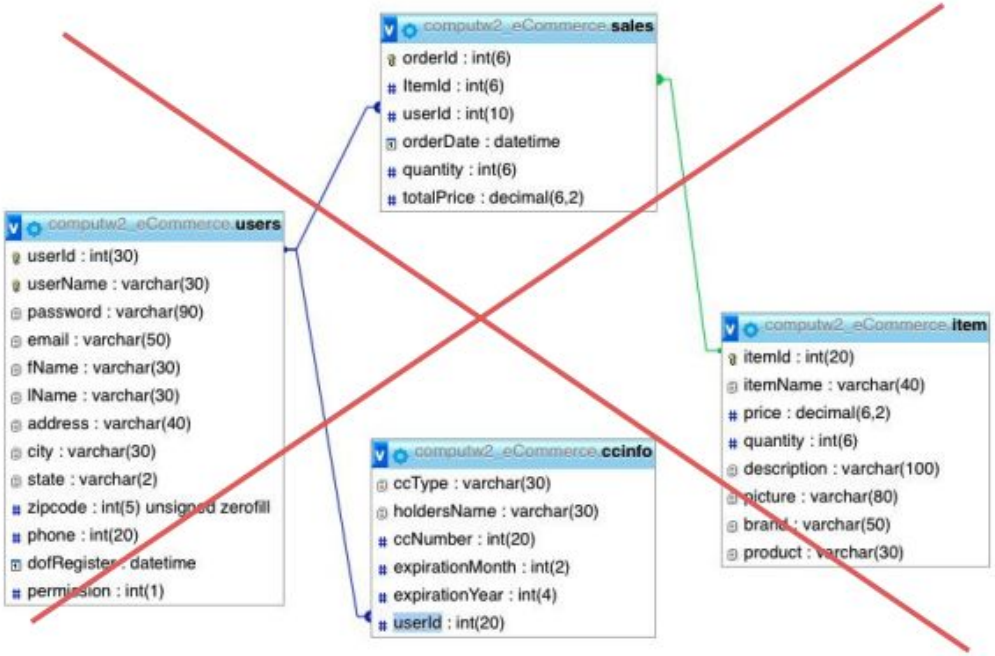
Legacy modernization

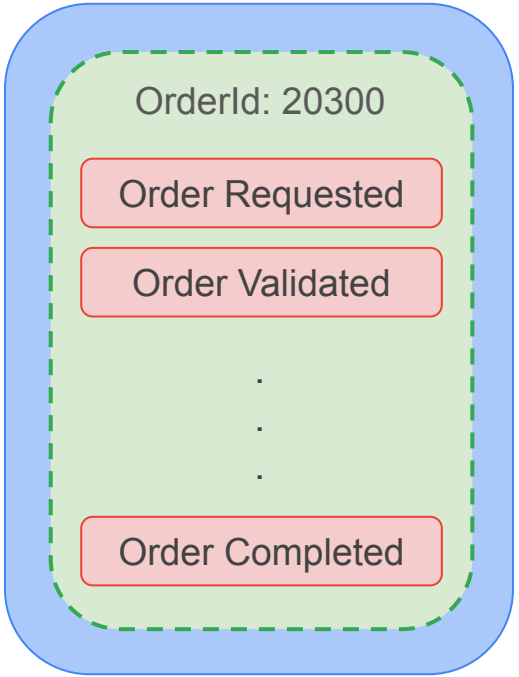
Escalando com snapshots

E por que perigoso?

Event Sourcing

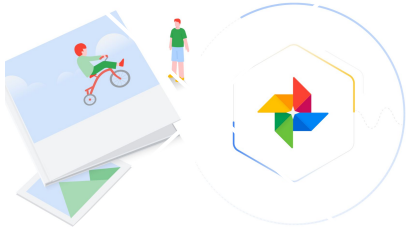
Eventos e não tabelas





Event Sourcing

Visibilidade dos dados

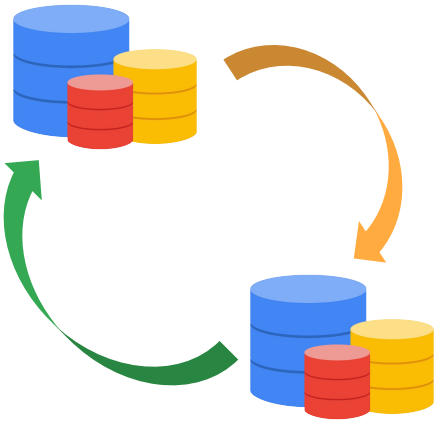


Escalando com snapshots

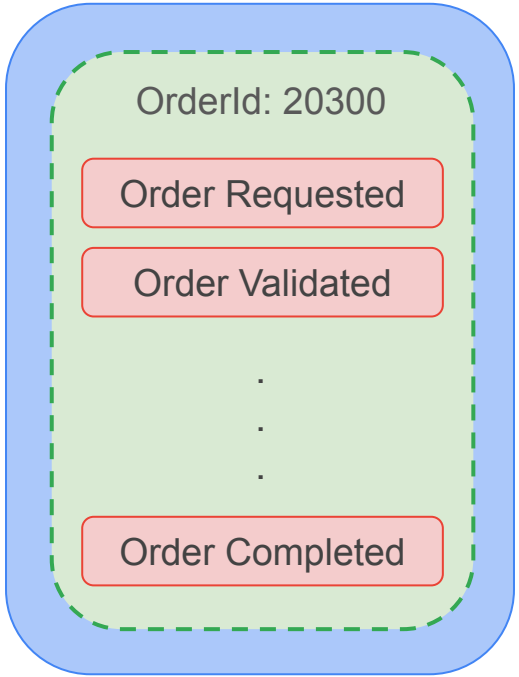
Event Sourcing

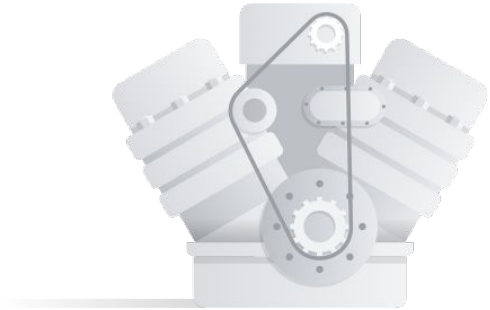
Schema changing

Database migration



Bilhões de registros

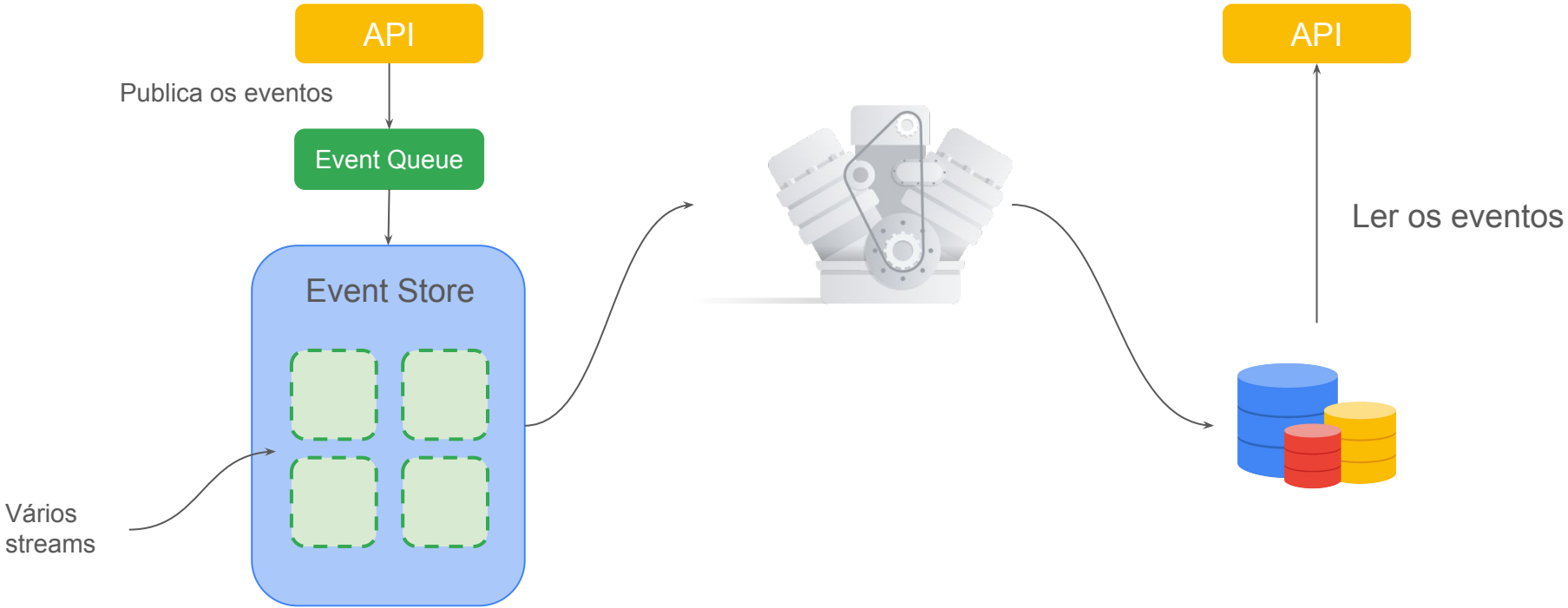




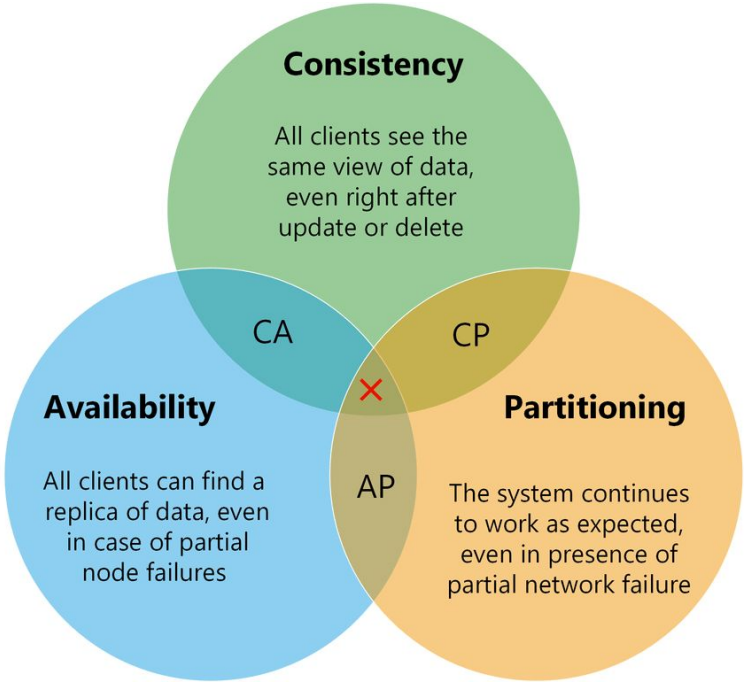
Eles não são imutáveis?

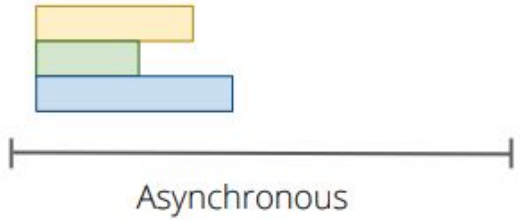
Event Sourcing

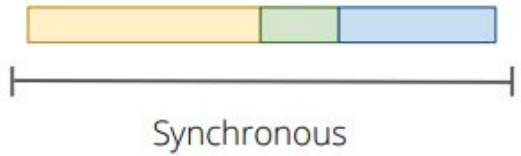
Lidando com domínios complexos



Eventual Consistency







Fonte: https://www.researchgate.net

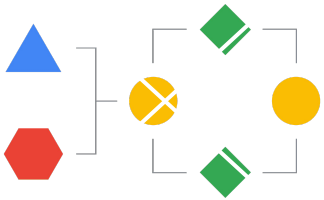
Event Sourcing

Reversing Events

Reversing Events

- **ADD** e não **SET**
- Armazenando diferenças nos eventos?
- Há uma consequência significativa quando a lógica de processamento está dentro de um modelo de domínio
  - O modelo de domínio pode alterar seu estado interno
  - Eles podem não ser visíveis para o processamento do objeto de evento
- Aplicar a partir do anterior mais recente e reproduzir o fluxo do evento

Event Sourcing

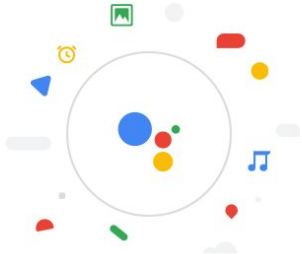


Code Changes

Novas features

Corrigir defeitos

Lógica temporal

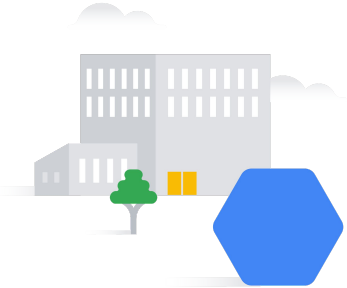


External Updates

Lidar com sistemas que seguem esta abordagem

Lidar com sistemas que **não** seguem esta abordagem

- Wrap o sistemas externos em um Gateway



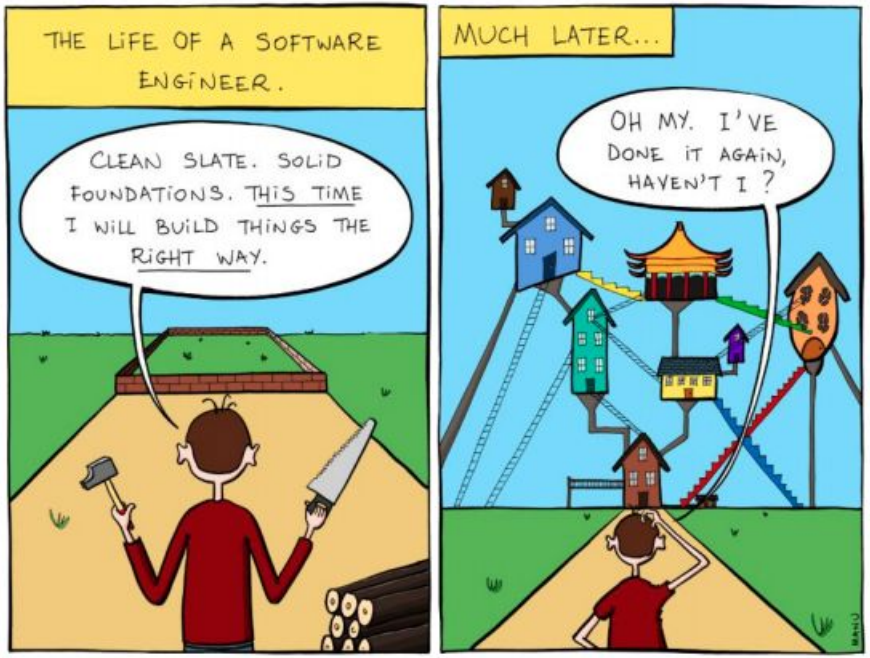
External Queries

Você confia em dados externos?

Praonquevô?

Event Sourcing

Coisas que aconteceram comigo em um passado nem tão distante




Ed Sheeran - Dive [Official Audio]
https://www.google.com
ALL
IMAGES
TV
LIVE
The Voice
NBA Basketball
College Basket
The Simpsons

Google

Quer brincar um pouquinho?

Event.Store

A simple and fast EventStore that support multiple persistence and notification providers

Event.Store

Event.Store is an open source library to create Event Stores that works with multiple persistence providers and notification systems.

The Event Store is a database accompanied by a publication and subscription system. The database stores all the events related to an event stream. The pub / sub system allows other systems or microservices to react to changes in event streams. It is a core component in any event sourcing + CQRS architectures.

Event.Store has client implementations for Node.js and Java. The two clients can interact with the same Event Store.

The Event Store support the folowing persistence providers:

And the folowing notification systems:

It is possible to create custom providers and publishers.

GitHub Node JS

GitHub Java

// Usage Guide

/// Create EventStore

To Create an EventStore you must provide two implementations:

- A persistence **Provider**: Responsible for events persistence in the store.
- A notification **Publisher** (Optional): Responsible for notify any process interested in modifications on the store streams.

If there is no publisher provided, the event store will not send any notification.

Javascript | Java

```
const eventStore = new EventStore(
    new InMemoryProvider(), // The persistence provider. Could use different
    new InMemoryPublisher()); // Opcional. Support different publishers, lik
```

/// Adding Events

To add Events you need to ask to EventStore a reference to an **EventStream**. You can add Events passing anything you want as a payload.

Javascript | Java

```
const ordersStream = eventStore.getEventStream('orders', '1234567');
ordersStream.addEvent({ data: 'My Event Payload'}); // Could pass anything
```

/// Reading to Events

To read Events you need to ask to EventStore a reference to an **EventStream**. You can read a stream to receive an ordered list containing all the events in the store.

Javascript | Java

```
const ordersStream = eventStore.getEventStream('orders', '1234567');
const events = await ordersStream.getEvents();
const order = ordersAggregation.loadFromHistory(events)
```

eventstore.net.br

Google

Event.Store

A simple and fast EventStore that support multiple persistence and notification providers

Event.Store

Event.Store is an open source library to create Event Stores that works with multiple persistence providers and notification systems.

The Event Store is a database accompanied by a publication and subscription system. The database stores all the events related to an event stream. The pub / sub system allows other systems or microservices to react to changes in event streams. It is a core component in any event sourcing + CQRS architectures.

Event.Store has client implementations for Node.js and Java. The two clients can interact with the same Event Store.

The Event Store support the folowing persistence providers:

And the folowing notification systems:

It is possible to create custom providers and publishers.

GitHub Node JS

GitHub Java

/// Reacting to Events

You can add subscribers to be notified every time a new event is added in any stream contained in the given aggregation. The following example will be notified for every new event in any stream in the "orders" aggregation.

Javascript | Java

```
eventStore.subscribe('orders', message => {
    console.log(message.aggregation);
    console.log(message.streamId);
    console.log(message.event.payload);
});
```

/// Removing subscriptions

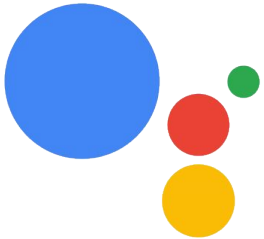
It is possible to cancel subscriptions to event stream channels.

Javascript | Java

```
const subscription = await eventStore.subscribe('orders', message => {
    console.log(message.aggregation);
    console.log(message.streamId);
    console.log(message.event.payload);
});
// ...
subscription.remove();
```

eventstore.net.br

Google



Obrigado