Lado negro das Arquiteturas Orientadas a Eventos

Rafael Girolineto

@rafapg

Tech Lead na Arquivei (temos vagas)
Opensanca
11 anos no mercado de dev
Eng Comp USP
Coffee Geek

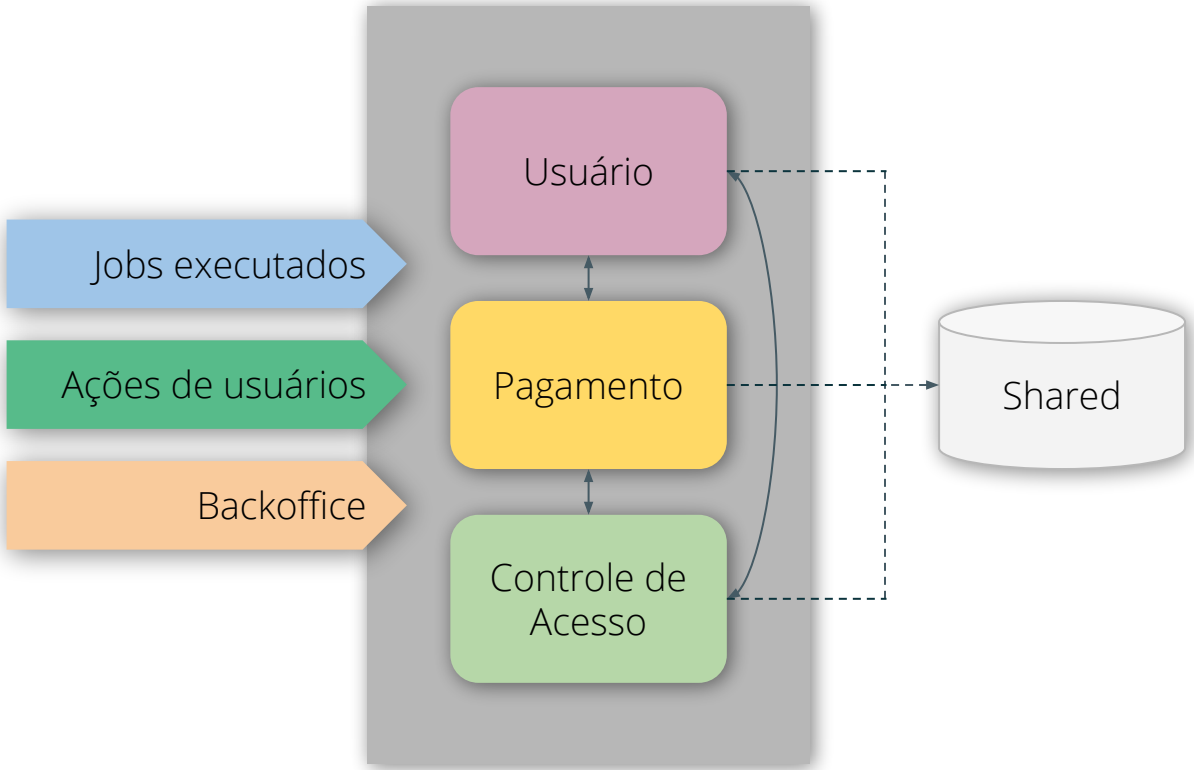
Sistema de consulta e armazenamento de Documentos Fiscais Eletrônicos (DF-e).

Ciclo de vida de um DFe é complexo e varia de documento em documento.

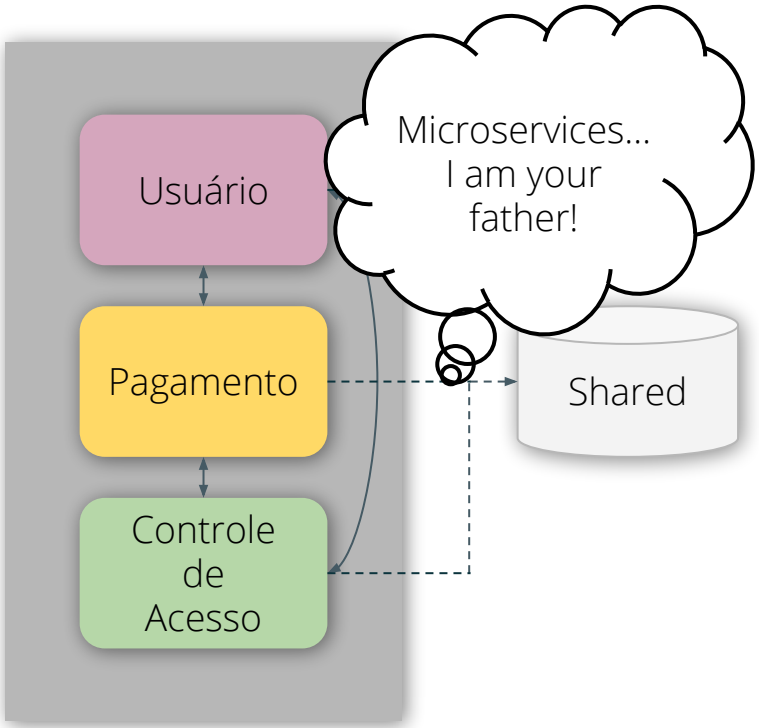
Arquitetura orientada a eventos ajuda a resolver esse problema

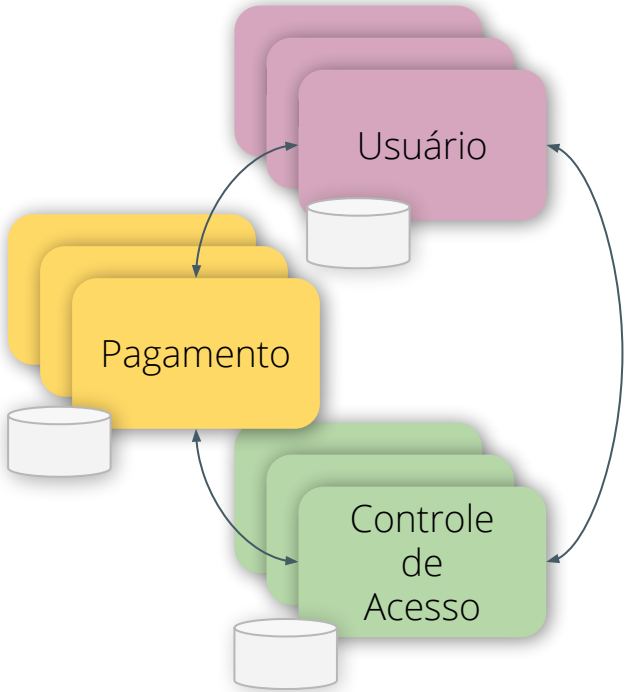
Mas o que é Arquitetura Orientada a Eventos?

O Poder do ~~Império~~ Monolito

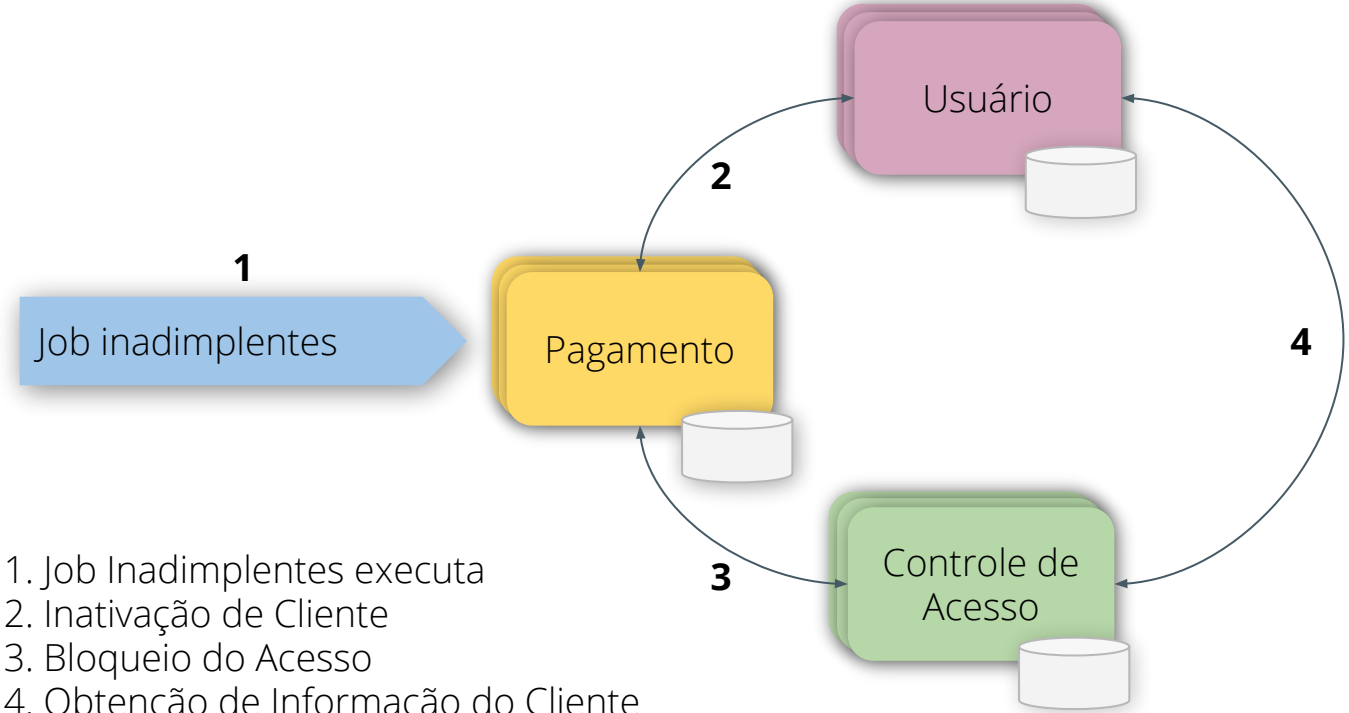


Monolito vs Microserviços





Integração entre serviços

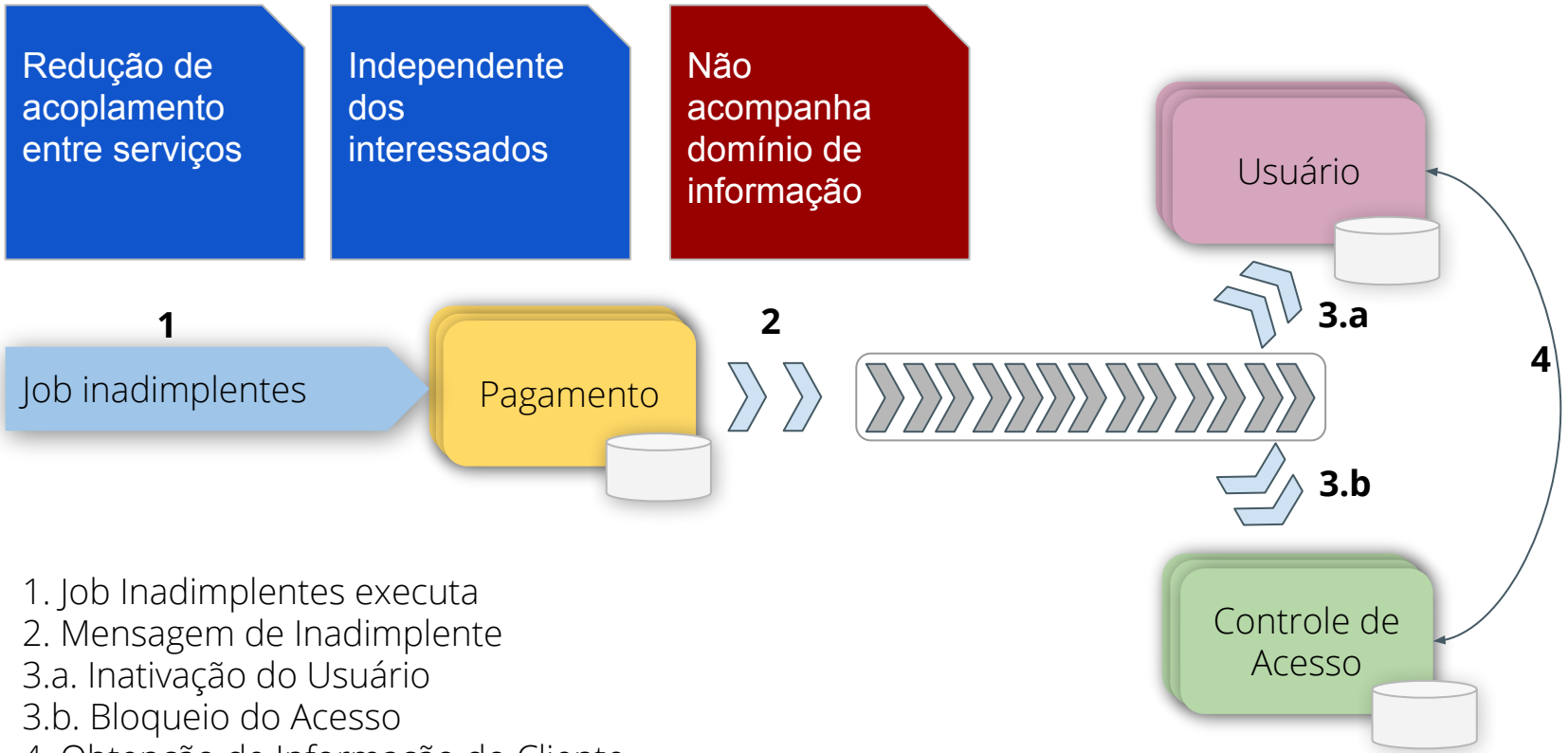


1. Job Inadimplentes executa
2. Inativação de Cliente
3. Bloqueio do Acesso
4. Obtenção de Informação do Cliente

Alto acoplamento entre serviços

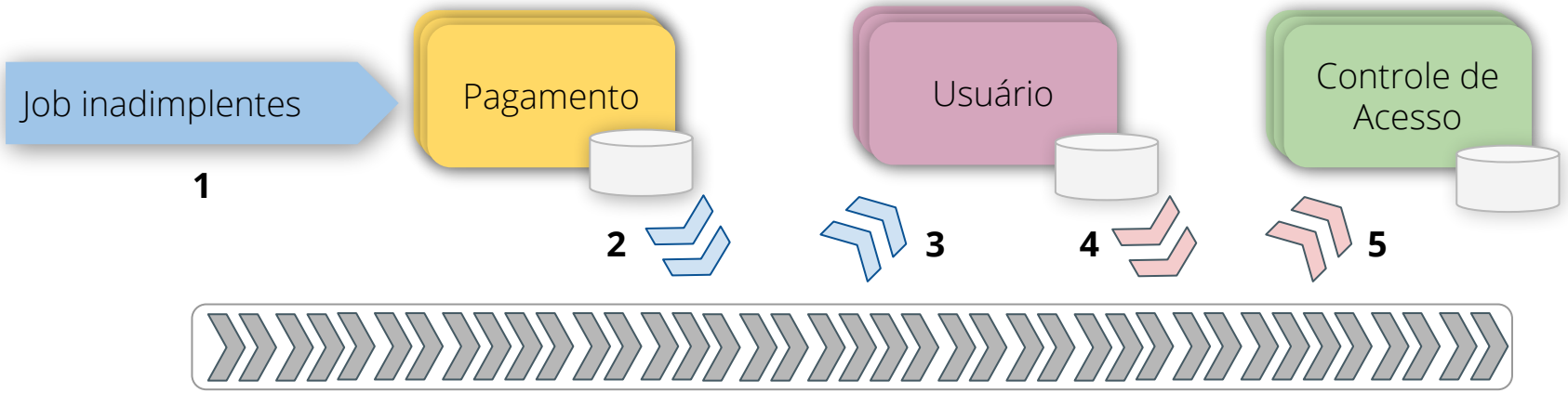
Conhecimento antecipado de todos os interessados

Mensagem / Evento de cliente inadimplente



1. Job Inadimplentes executa
2. Mensagem de Inadimplente
3.a. Inativação do Usuário
3.b. Bloqueio do Acesso
4. Obtenção de Informação do Cliente

Stream de Eventos

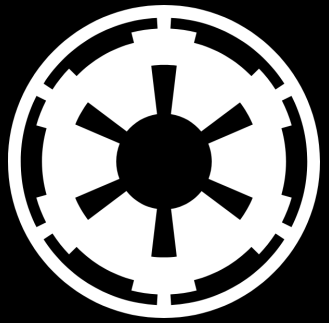


1. Job Inadimplentes executa
2. Evento de Inadimplente
3. Inativação do Usuário
4. Evento de Usuário Inativo
5. Bloqueio do Acesso

- Redução de acoplamento entre serviços
- Independente dos interessados
- Acompanha domínio de informação

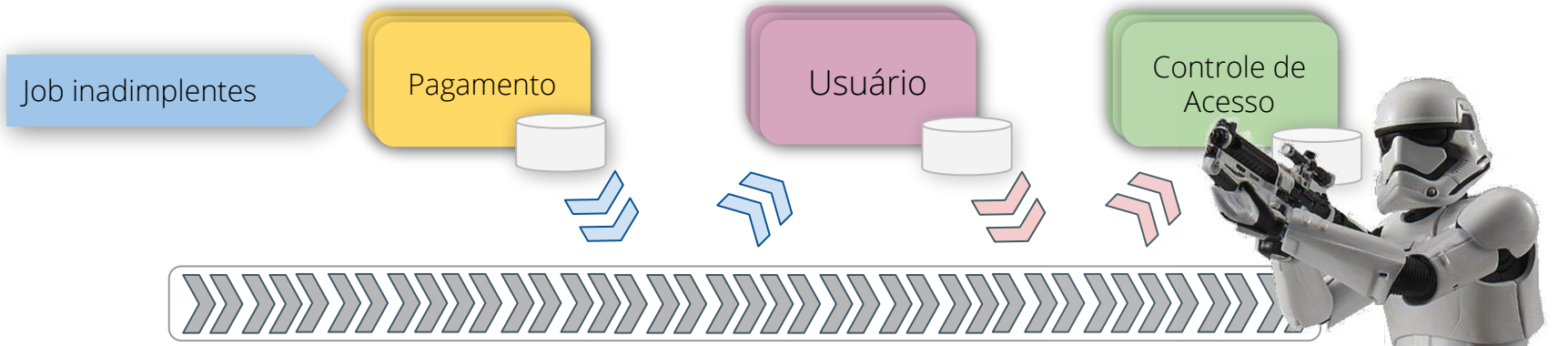
Vantagens

- Diminuição do acoplamento
- Processamento assíncrono
- Alta escalabilidade
- Limitação do escopo dos serviços



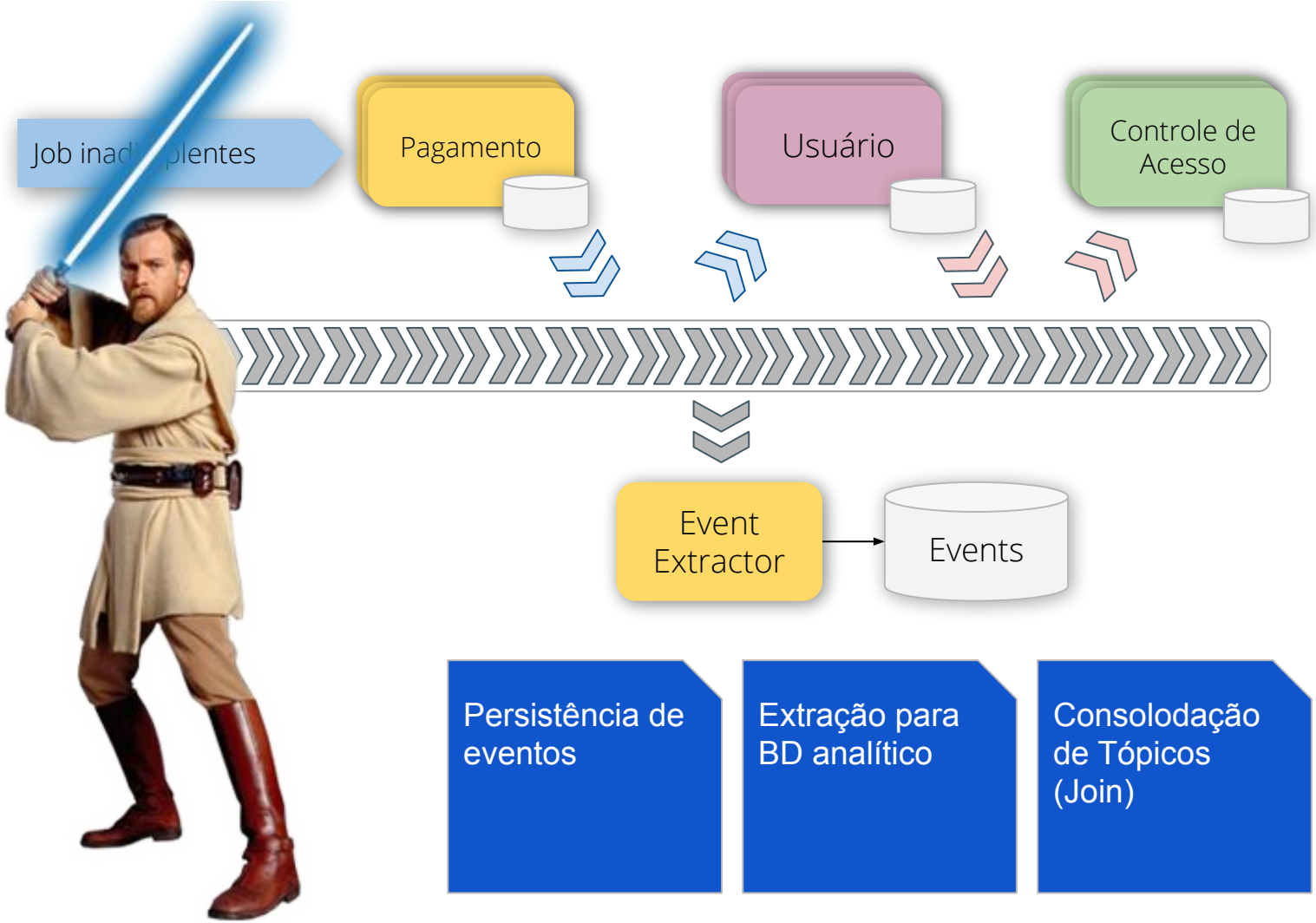
Onde isso pode dar errado?

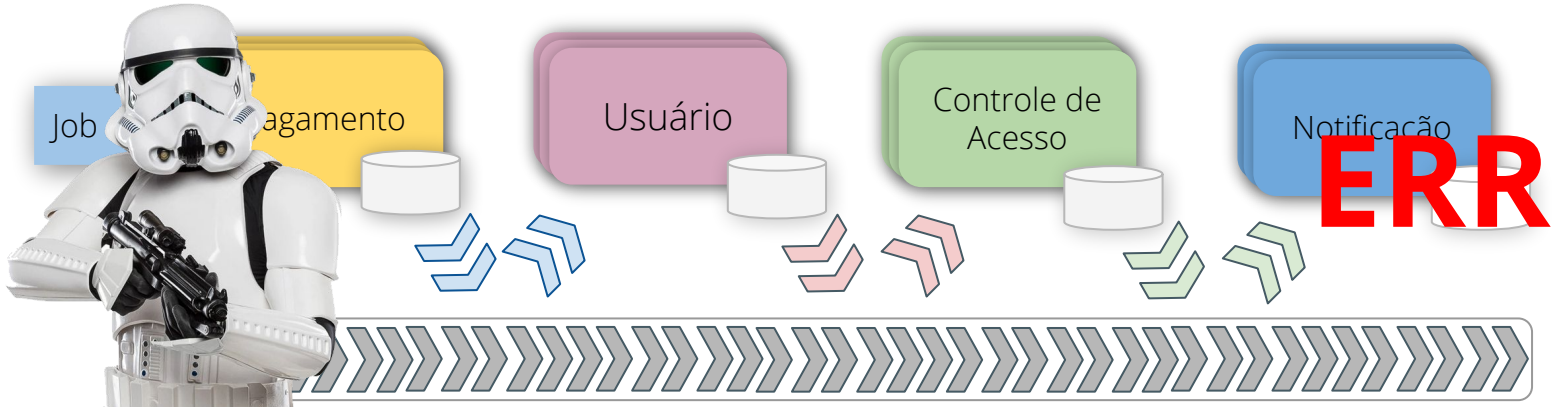
Como resolver esses problemas?



Como gerar um relatório de usuários bloqueados por inadimplência?

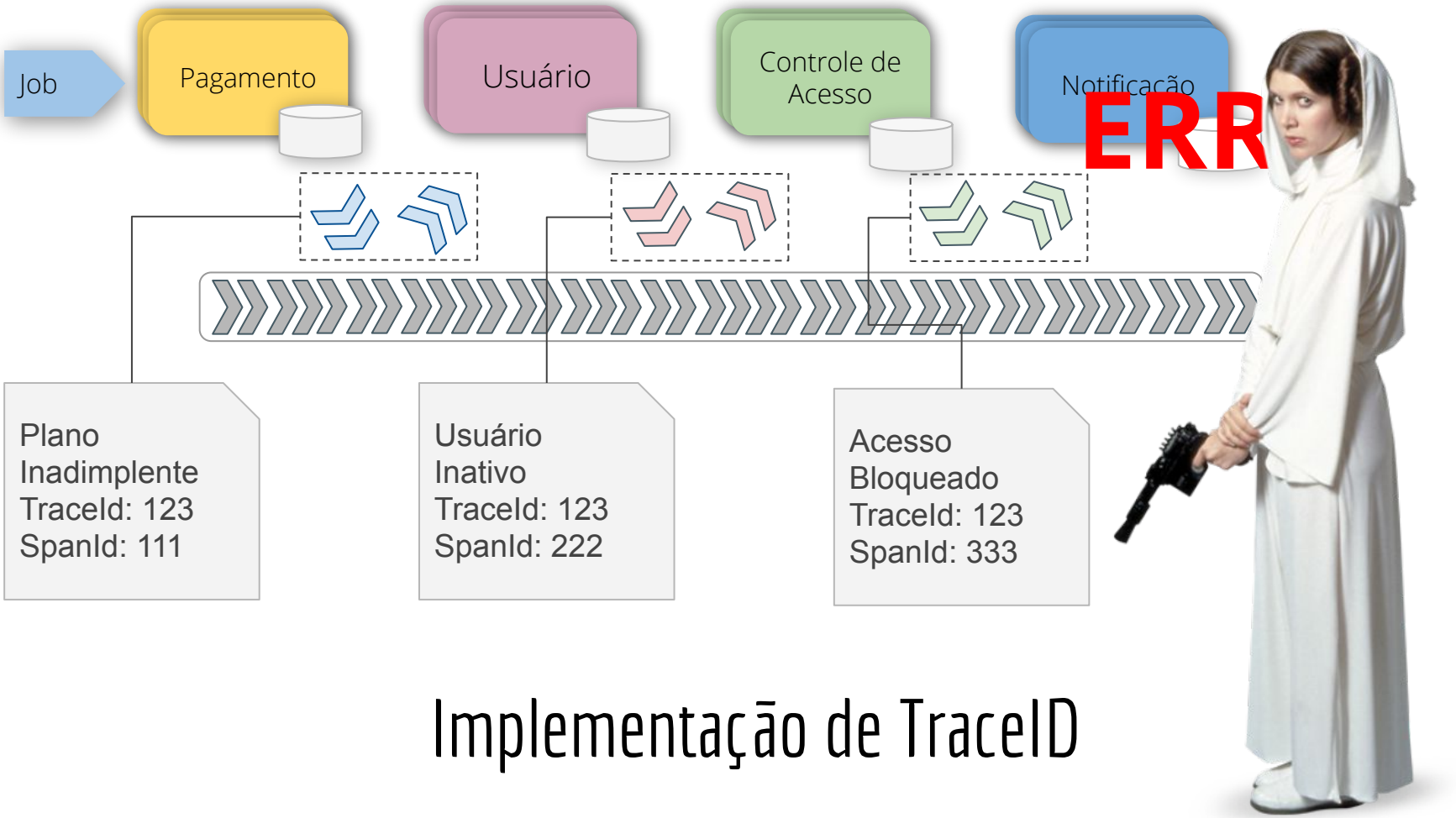
Falta visão analítica consolidada


Job inadi plentes
Pagamento
Usuário
Controle de Acesso
Event Extractor
Events
Persistência de eventos
Extração para BD analítico
Consolodação de Tópicos (Join)

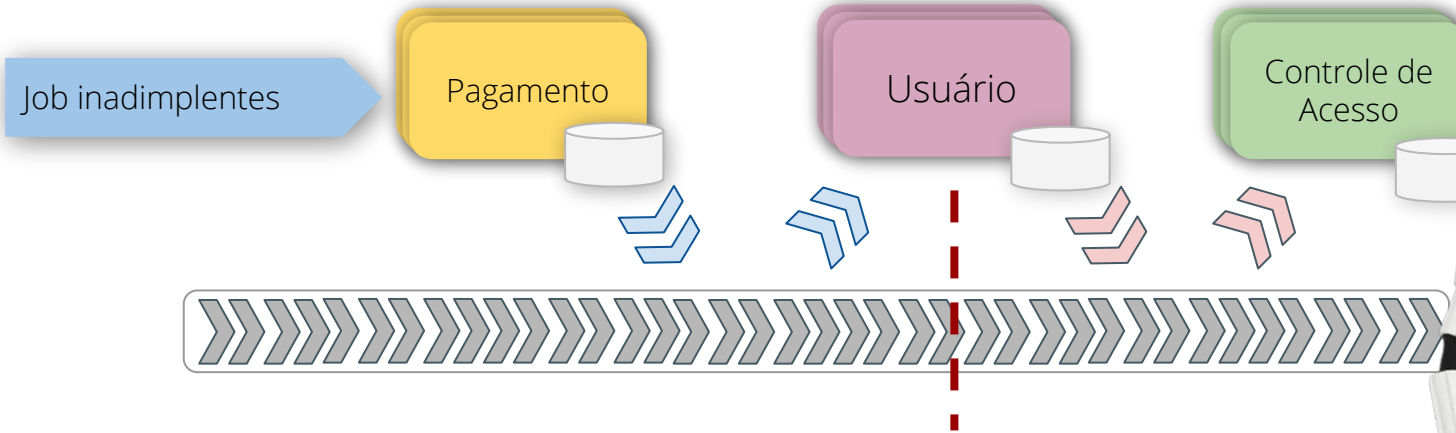


Como descobrir o que originalmente levou ao erro na notificação?

Baixa Rasterabilidade



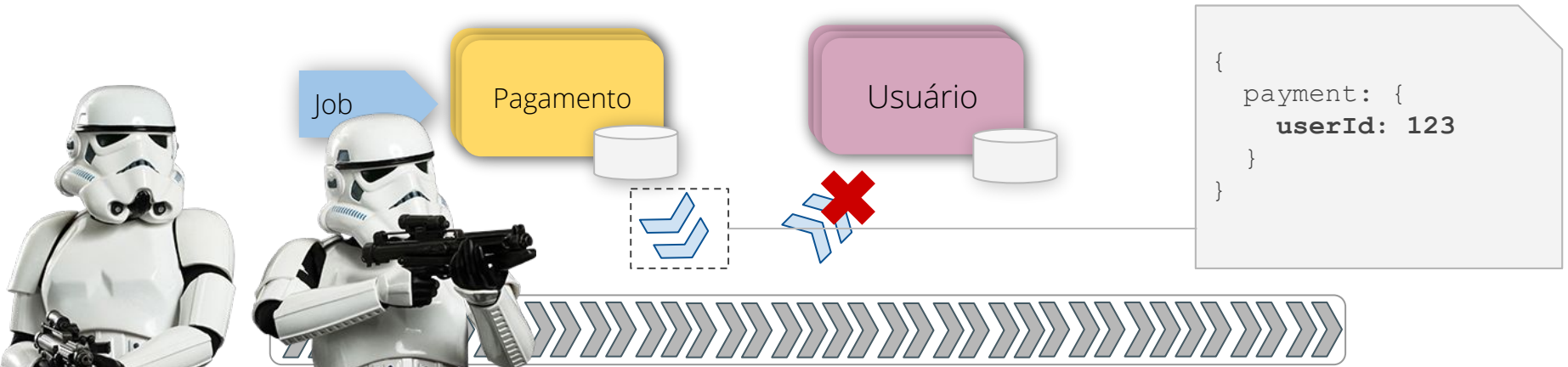
Implementação de TraceID



Em um determinado momento o usuário inativo é capaz de acessar o sistema

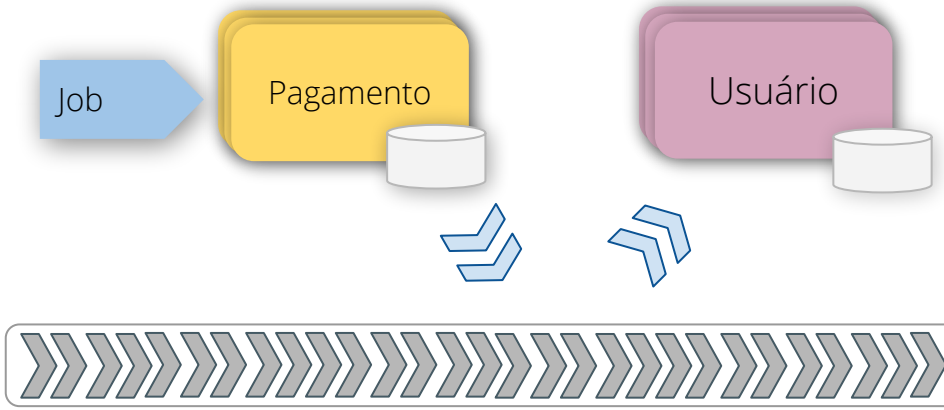
Consistência Eventual

Projetar Considerando Consistência Eventual



Serviço de usuário quebra na mudança do formato do evento de inadimplência

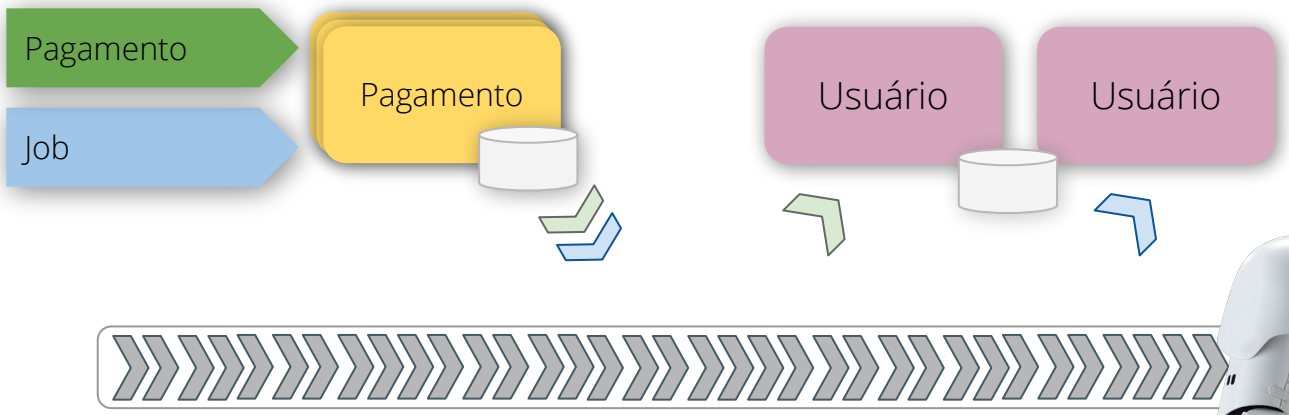
Quebra de Contrato




```
{
  version: 1,
  userId: 123,
  payment: {
  }
}
```

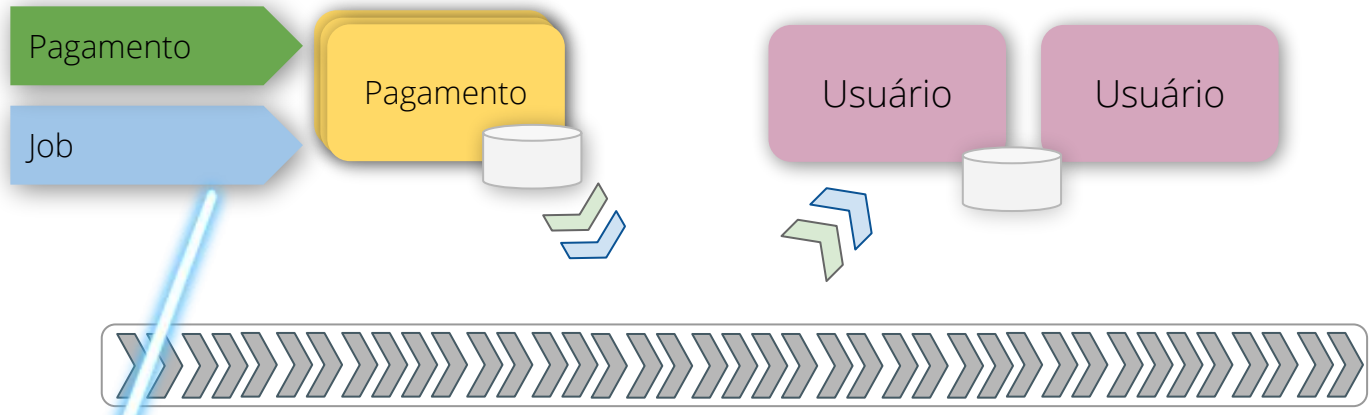
```
{
  version: 1,
  payment: {
    userId: 123
  }
}
```

Validação de Schema



Eventos de pagamento processados em ordem invertida.
Cliente ficará bloqueado mesmo após pagar

Processamento Fora de Ordem



Cada instância responde por uma parcela dos usuários

A ordem é respeitada dentro da partição

Particionamento


Quem irá prevalecer?

Quando Evitar?

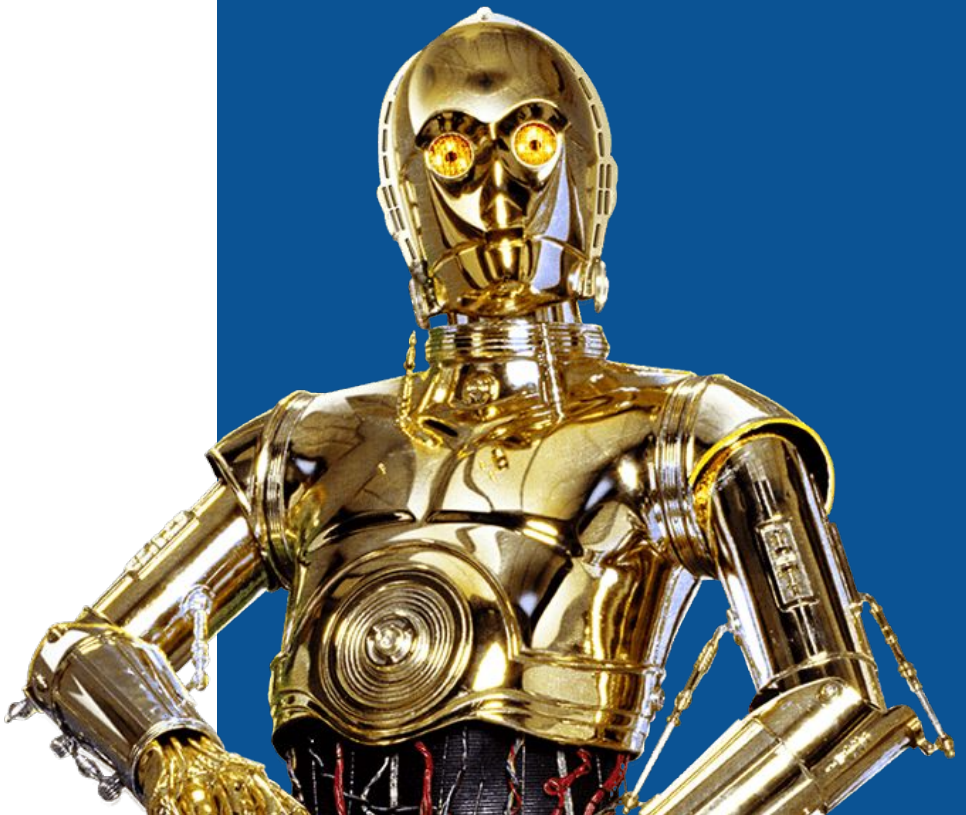
- Se a consistência é primordial
- Quando a ordem é muito importante e não é possível particionar
- Não existe rastreabilidade entre eventos

Quando Adotar?

- Processamento síncrono é importante
- Queremos desacoplar os contextos
- Aceitamos a consistência eventual
- Temos processos para lidar com as divergências

MAY THE ~~FORCE~~ *event*
BE
WITH YOU

Perguntas?



Obrigado!!!

@rafapg

Vantagens

1. Diminuição do acoplamento
2. Processamento assíncrono
3. Alta escalabilidade
4. Limitação do escopo dos serviços

Problemas

- Falta de visão analítica / consolidada
- Baixa rastreabilidade
- Consistência Eventual
- Quebra de contrato
- Processamento fora de ordem

Como resolver esses problemas?

Falta de visão analítica / consolidada

1. Processo de consolidação verificou inadimplência
2. Evento de inadimplência inativou o usuário
3. Inativação do usuário suspendeu o acesso

Como descobrir quais casos de suspensão de acesso por inadimplência?

Falta de visão analítica / consolidada

- Armazenamento dos eventos
- Extração dos dados de evento para BD analíticos
- Join entre tópicos (Kafka Streams)

Baixa rastreabilidade

1. Processo de consolidação verificou inadimplência
2. Evento de inadimplência inativou o usuário
3. Inativação do usuário suspendeu o acesso
4. Suspensão do acesso disparou uma notificação com erro

O que originou inicialmente a notificação com erro?

Baixa rastreabilidade

- Implementação de tracing distribuído (OpenTracing)
  - Logs
  - Controle de TraceId

Consistência Eventual

1. Processo de consolidação verificou inadimplência
2. Evento de inadimplência inativou o usuário
3. Inativação do usuário suspendeu o acesso

Existe um instante onde o usuário estará inativo e inadimplente, mas poderá acessar o sistema

Consistência Eventual

- Gestão de estados "sujos”
- Uso de processamento síncrono em casos críticos
- Aceita que dói menos

Rafael Cardoso

@rafapg

Tech Lead na Arquivei (temos vagas)
11 anos no mercado de dev
Eng Comp 04 USP São Carlos
Coffee Geek (coffee chato)

Quebra de contrato

1. Serviço de pagamento altera o contrato de evento de inadimplência
2. Serviço de usuário não consegue processar evento

Toda a cadeia de eventos será impactada

Quebra de contrato

- Implementação de validação de schema (Avro)
- Produtor e consumidor compartilham o mesmo contrato

Processamento fora de ordem

1. Consolidação verificou inadimplência gerando evento de inadimplência
2. Cliente realiza pagamento gerando evento de adimplência
3. Serviço de usuário recebe eventos em ordem invertida

Cliente adimplente será inativado

Processamento fora de ordem

- Garantia de ordem / Particionamento
- Processamento síncrono para casos críticos
- Persistência de informação não consolidada (data de suspensão e data de pagamento)

THE FORCE IS STRONG WITH THIS ONE