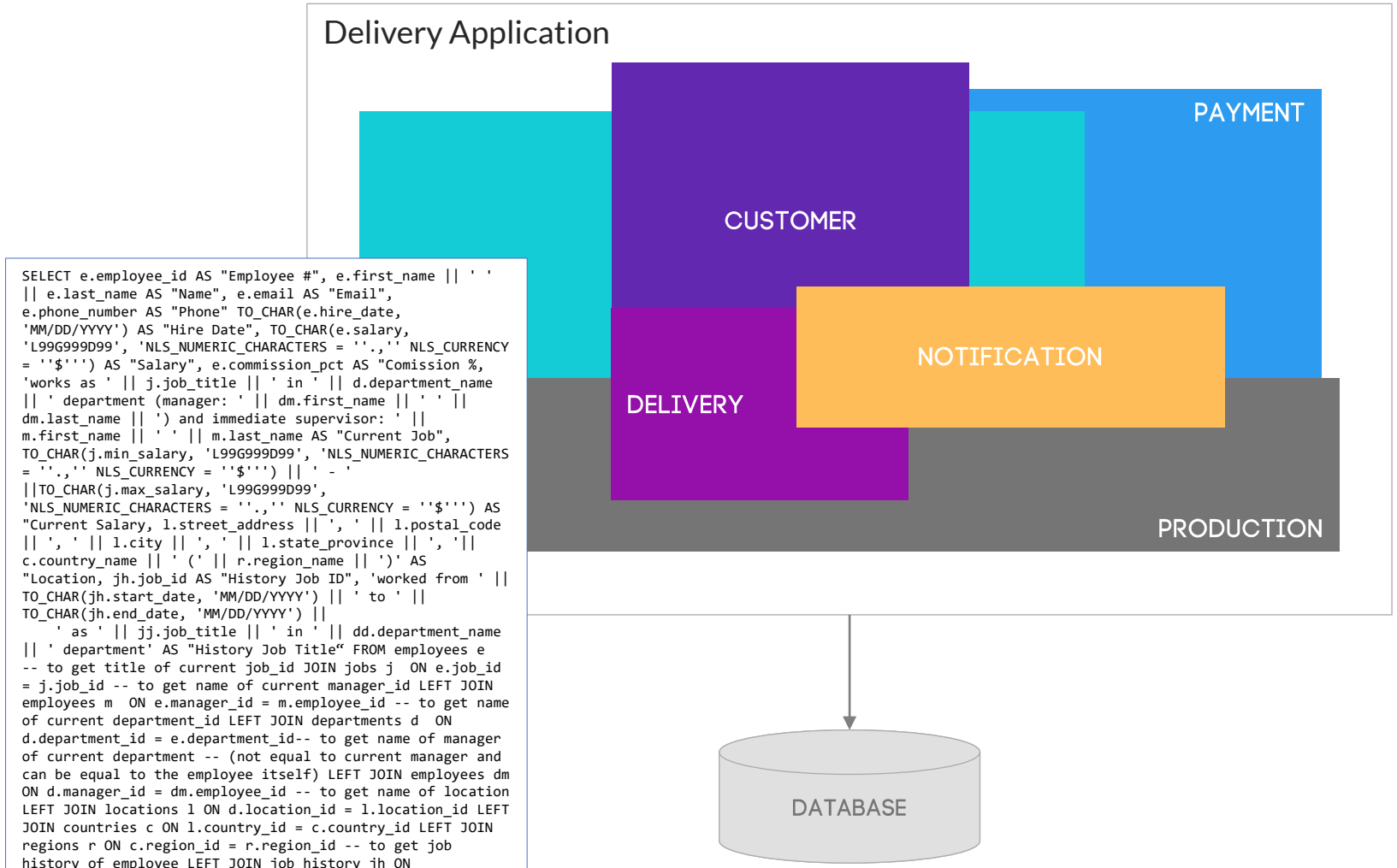

19/10
PROJECT Decoder
Fundamentos Microservices
Desafios e Equívocos

O que você verá:

- Como a Arquitetura de Microservices veio para solucionar as complexidades dos negócios modernos;
- Os fundamentos de Microservices;
- Princípios e equívocos que envolvem esse modelo arquitetural;
- Sincronia de dados em sistemas distribuídos;
- Consistência Eventual e o Teorema CAP;
- Identificadores distribuídos;
- Questão do acoplamento e disponibilidade em Arquitetura de Microservices;
- Distribuição das base de dados em Microservices;
- Concepção e utilização de Microservices;
- Algumas premissas importantes sobre Microservices;
- E muito mais....

Michelli Brito
@brito_michelli

Arquitetura Monolítica




```
SELECT e.employee_id AS "Employee #", e.first_name || ' '
|| e.last_name AS "Name", e.email AS "Email",
e.phone_number AS "Phone" TO_CHAR(e.hire_date,
'MM/DD/YYYY') AS "Hire Date", TO_CHAR(e.salary,
'L99G999D99', 'NLS_NUMERIC_CHARACTERS = ''.,'' NLS_CURRENCY
= ''$''') AS "Salary", e.commission_pct AS "Comission %,
'works as ' || j.job_title || ' in ' || d.department_name
|| ' department (manager: ' || dm.first_name || ' ' ||
dm.last_name || ') and immediate supervisor: ' ||
m.first_name || ' ' || m.last_name AS "Current Job",
TO_CHAR(j.min_salary, 'L99G999D99', 'NLS_NUMERIC_CHARACTERS
= ''.,'' NLS_CURRENCY = ''$''') || ' - '
||TO_CHAR(j.max_salary, 'L99G999D99',
'NLS_NUMERIC_CHARACTERS = ''.,'' NLS_CURRENCY = ''$''') AS
"Current Salary, l.street_address || ', ' || l.postal_code
|| ', ' || l.city || ', ' || l.state_province || ', '||
c.country_name || ' (' || r.region_name || ')' AS
"Location, jh.job_id AS "History Job ID", 'worked from ' ||
TO_CHAR(jh.start_date, 'MM/DD/YYYY') || ' to ' ||
TO_CHAR(jh.end_date, 'MM/DD/YYYY') ||
    ' as ' || jj.job_title || ' in ' || dd.department_name
|| ' department' AS "History Job Title“ FROM employees e
-- to get title of current job_id JOIN jobs j  ON e.job_id
= j.job_id -- to get name of current manager_id LEFT JOIN
employees m  ON e.manager_id = m.employee_id -- to get name
of current department_id LEFT JOIN departments d  ON
d.department_id = e.department_id-- to get name of manager
of current department -- (not equal to current manager and
can be equal to the employee itself) LEFT JOIN employees dm
ON d.manager_id = dm.employee_id -- to get name of location
LEFT JOIN locations l ON d.location_id = l.location_id LEFT
JOIN countries c ON l.country_id = c.country_id LEFT JOIN
regions r ON c.region_id = r.region_id -- to get job
history of employee LEFT JOIN job_history jh ON
e.employee_id = jh.employee_id -- to get title of job
history job_id LEFT JOIN jobs jj ON jj.job_id = jh.job_id -
- to get namee of department from job history LEFT JOIN
```

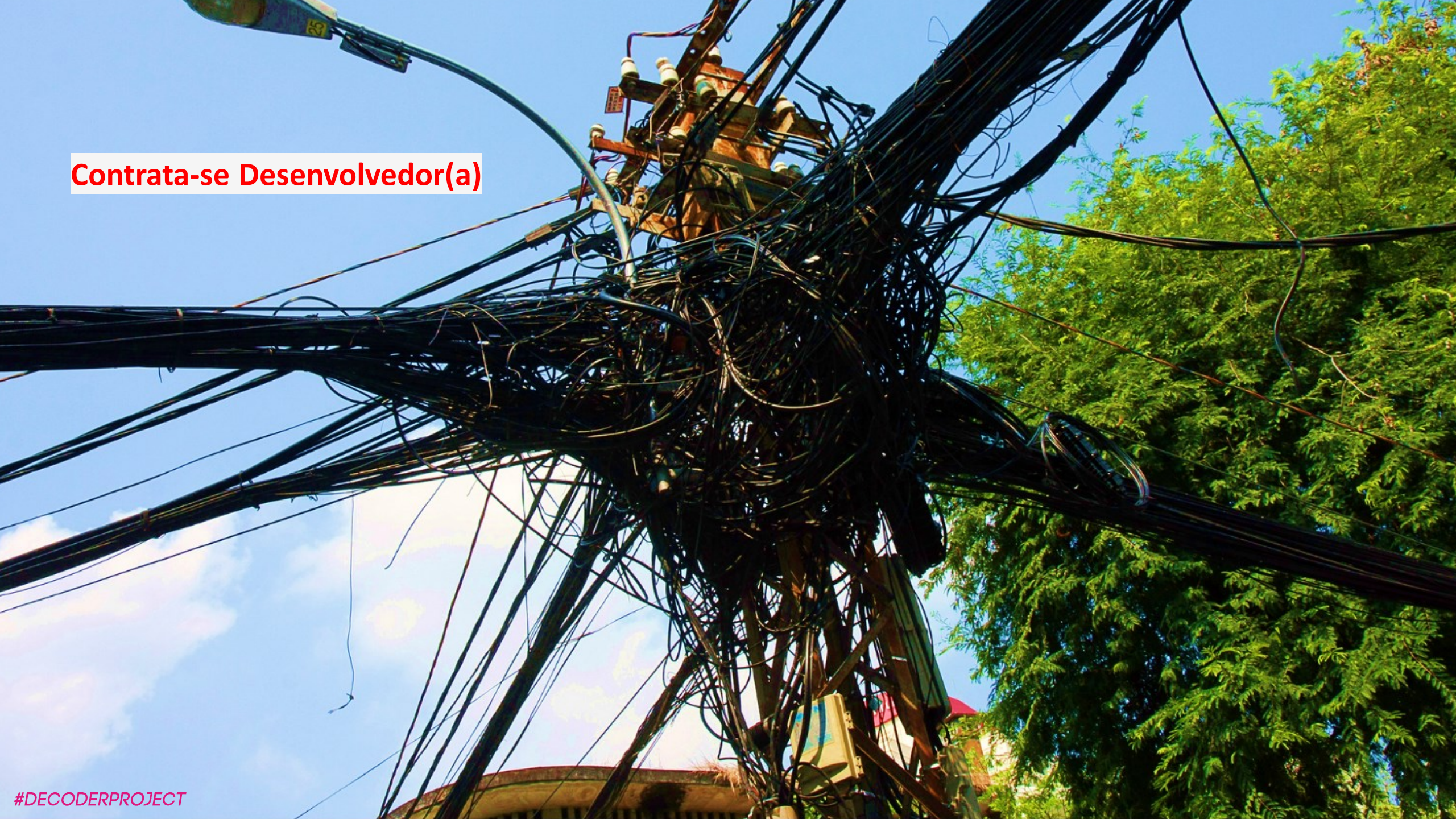
#DECODERPROJECT

Michelli Brito
@brito_michelli

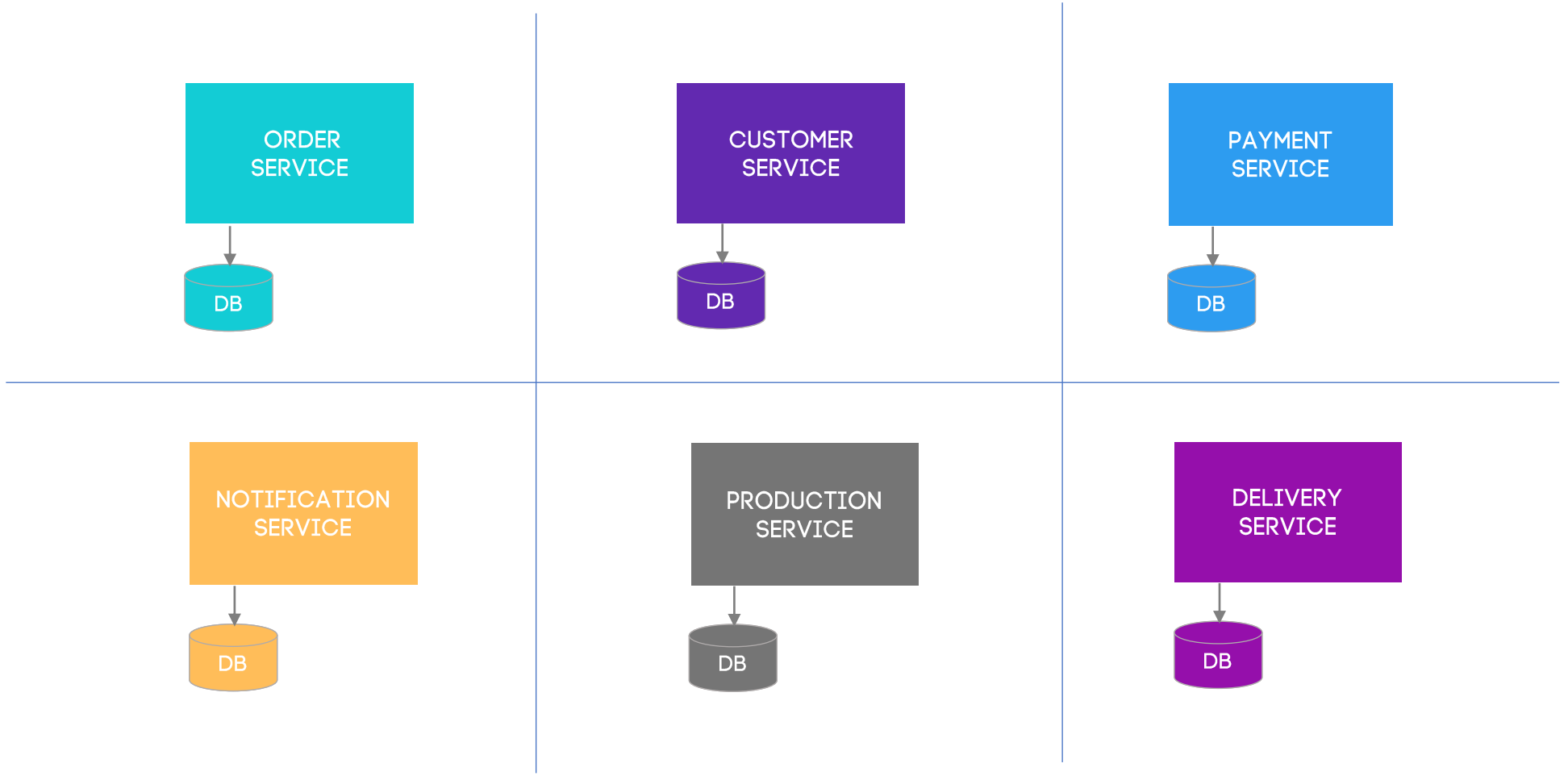
Expectativa

Realidade

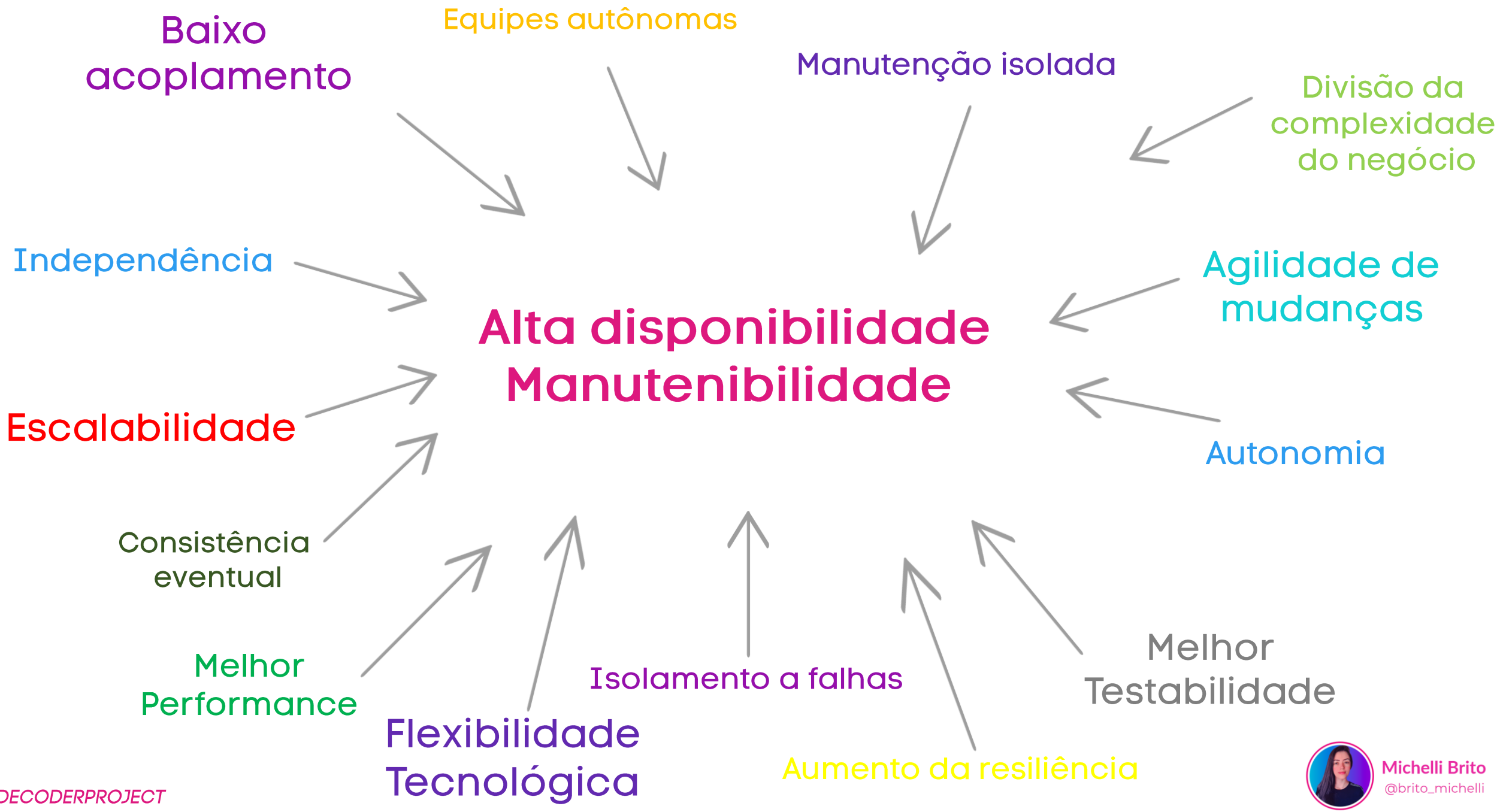

Só precisamos de um ajuste!
Arquiteto(a)
Monolito


Contrata-se Desenvolvedor(a)
#DECODERPROJECT

Arquitetura de Microservices



Michelli Brito
@brito_michelli



#DECODERPROJECT

Michelli Brito
@brito_michelli

Fundamentos de Microservices

Arquitetura de Microservices envolve desde conceitos técnicos, financeiros e gerenciais moldados principalmente sobre a necessidade do negócio.

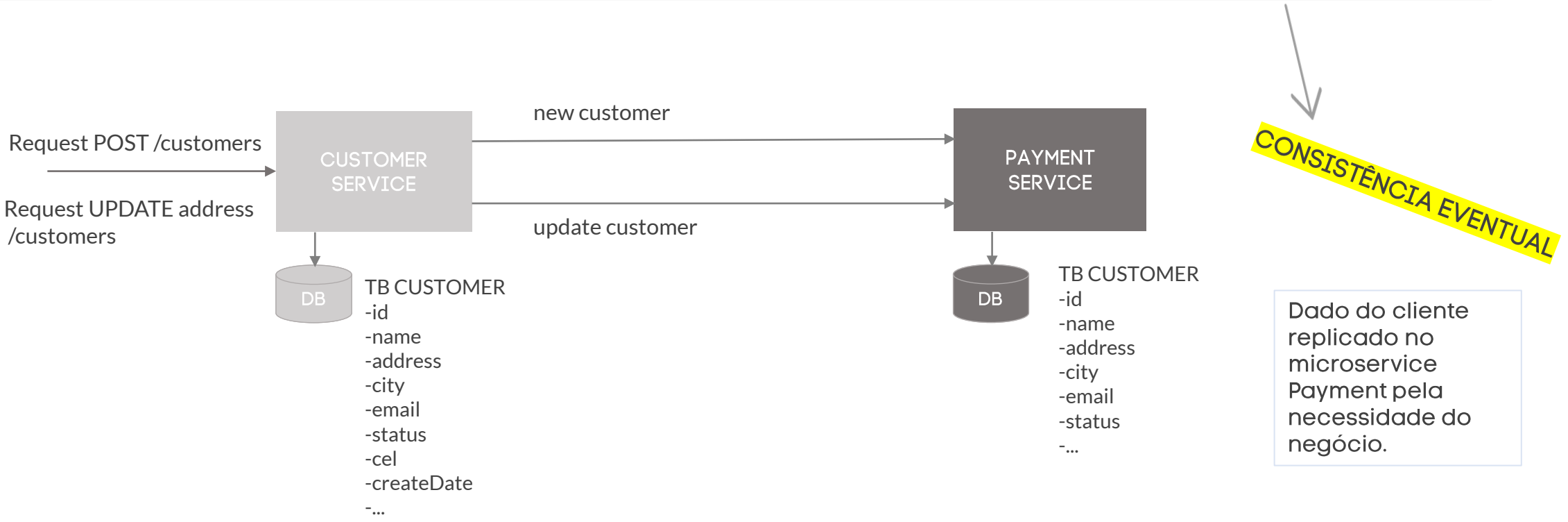
A mudança de paradigma para Microservices requer também uma mudança equivalente na estrutura da organização. Não é só uma questão técnica. É também cultural e organizacional.

Arquitetura de Microservices é um subconjunto dos conceitos de sistemas distribuídos modernos.

Michelli Brito
@brito_michelli

Sincronia de Dados em Sistemas Distribuídos

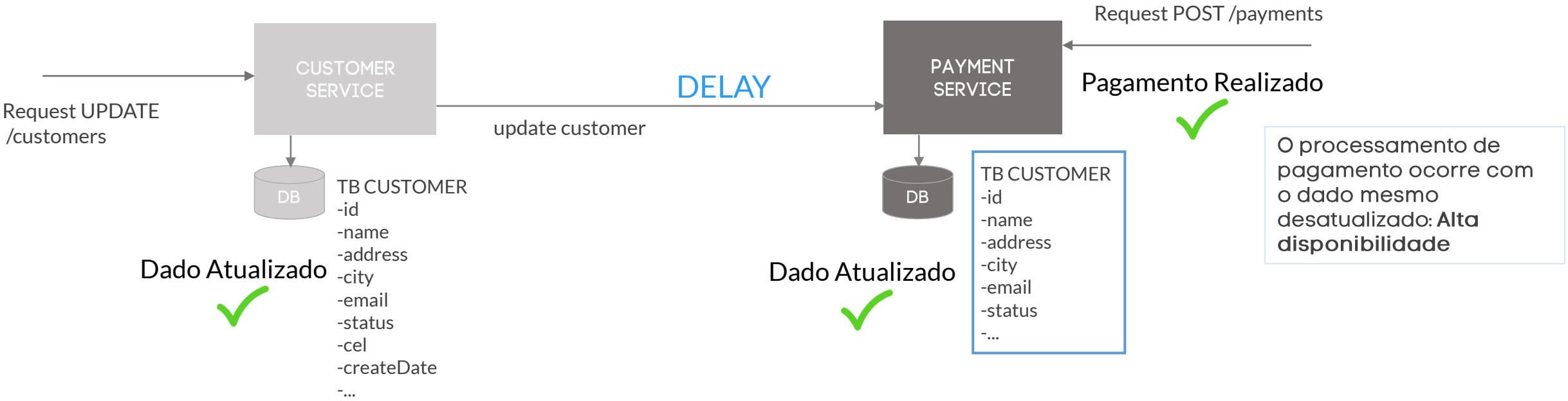
Não há como garantir alta disponibilidade com consistência forte ao mesmo tempo em uma arquitetura de Microservices com dados distribuídos.



Michelli Brito
@brito_michelli

Disponibilidade vs Consistência Forte : Teorema CAP

Para garantir a **alta disponibilidade** em sistemas distribuídos com dados compartilhados precisamos que a sincronia de dados ocorra de maneira assíncrona e não bloqueante, causando assim uma **consistência eventual,** onde os dados podem estar momentaneamente desatualizados ou ainda não replicados quando consultados.



Priorizando a disponibilidade : consistência eventual.

Michelli Brito
@brito_michelli

Identificadores UUIDs

IDs do tipo UUID são identificadores temporais universalmente exclusivos e essenciais para sincronia de dados distribuídos.

IDs Sequenciais	UUIDs Temporais
20	**68d3b1da-edca-4e38-a3d9-0c3e6f42f112**

podem ser gerados em qualquer lugar

garantem maior manutebilidade

facilita replicação de dados

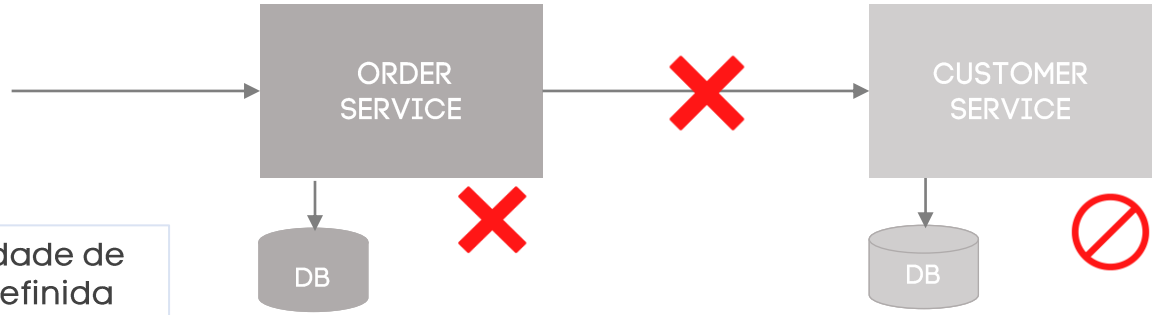
únicos em qualquer base de dados

Michelli Brito
@brito_michelli

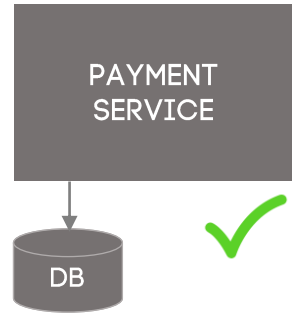
Acoplamento em Microservices

Não existe desacoplamento absoluto entre Microservices.

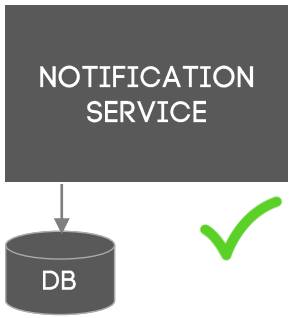
O forte acoplamento nem sempre é um problema de modelagem arquitetural, mas sim pode ser uma necessidade do próprio negócio.

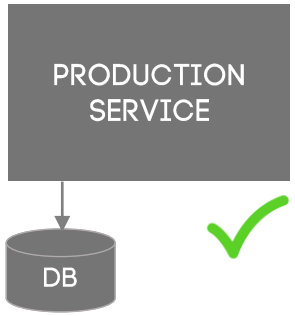


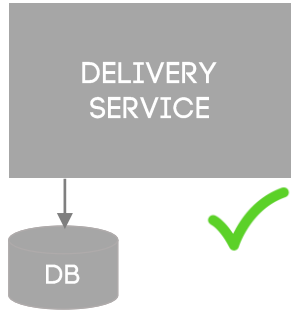
A necessidade de negócio definida obriga uma forte dependência entre Order e Customer.



O entendimento do negócio é essencial na definição da arquitetura.







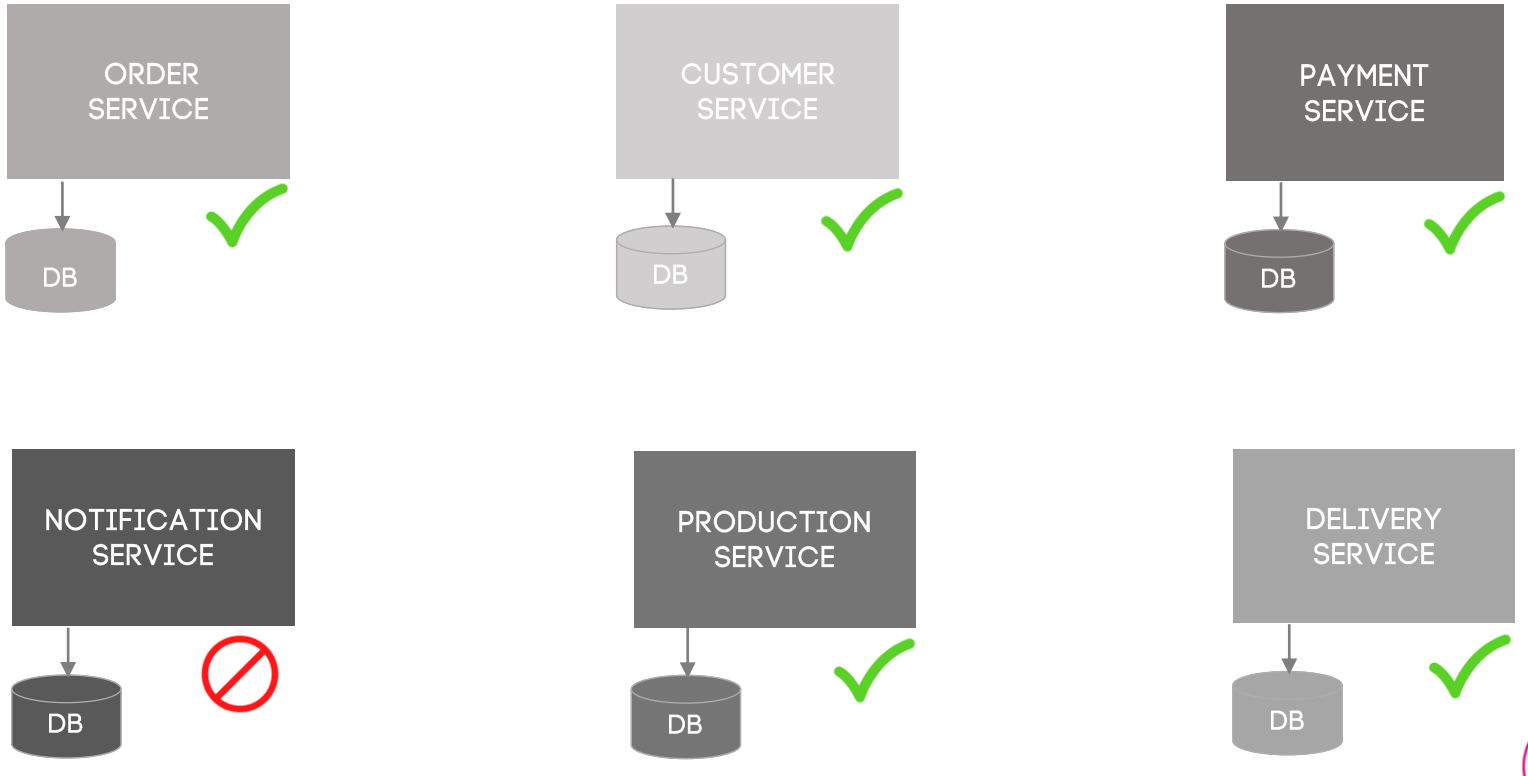
#DECODERPROJECT

Michelli Brito
@brito_michelli

Disponibilidade em Microservices

Não criamos Microservices para que qualquer um possa parar em algum momento sem afetar os demais, mas sim **para que alguns possam parar eventualmente e o sistema continuar disponível**. E isso já é muito melhor do que se nenhum pudesse parar.

A busca é sempre pela maior disponibilidade, mesmo não existindo na prática uma disponibilidade de 100%

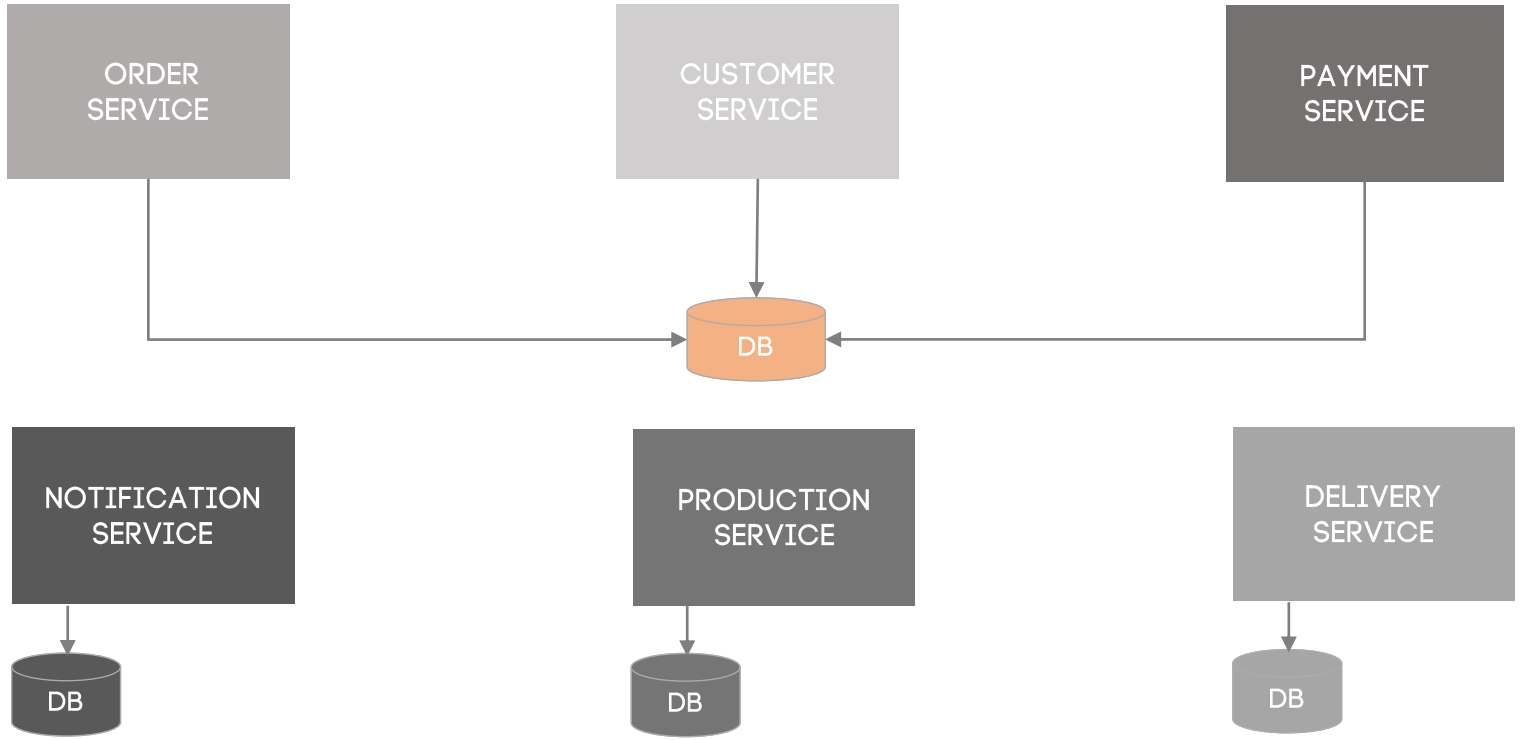


Michelli Brito
@brito_michelli

Distribuição das Bases de Dados em Microservices

Sim! Ainda temos uma Arquitetura de Microservices com base de dados compartilhada.

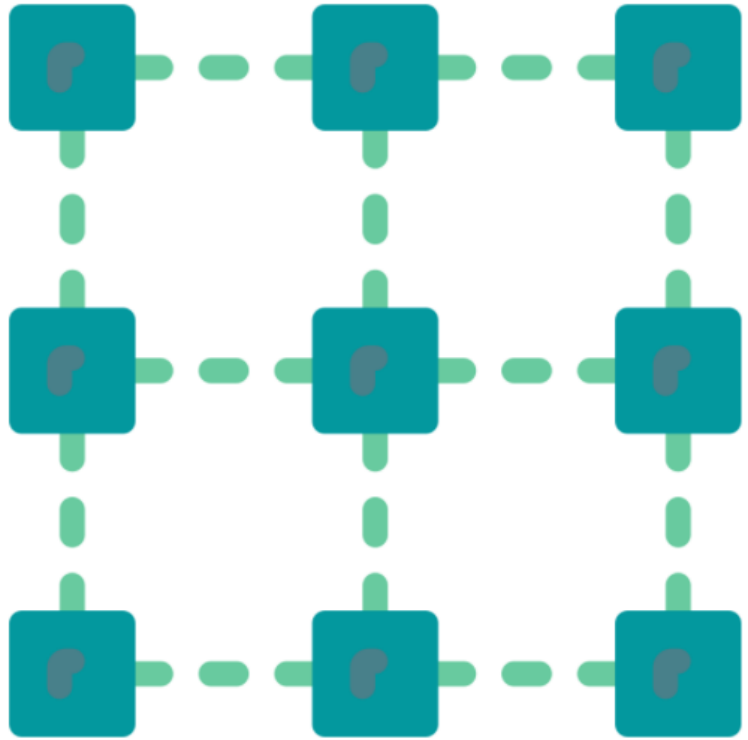
Na migração de Monolítico para Microservices o uso de base de dados compartilhada é comum no inicio.



Michelli Brito
@brito_michelli

Concepção e Utilização da Arquitetura de Microservices

Arquitetura de Microservices são conceitos e princípios que devem ser sempre adaptados ao negócio, não é um modelo e nem um padrão pronto e replicável.

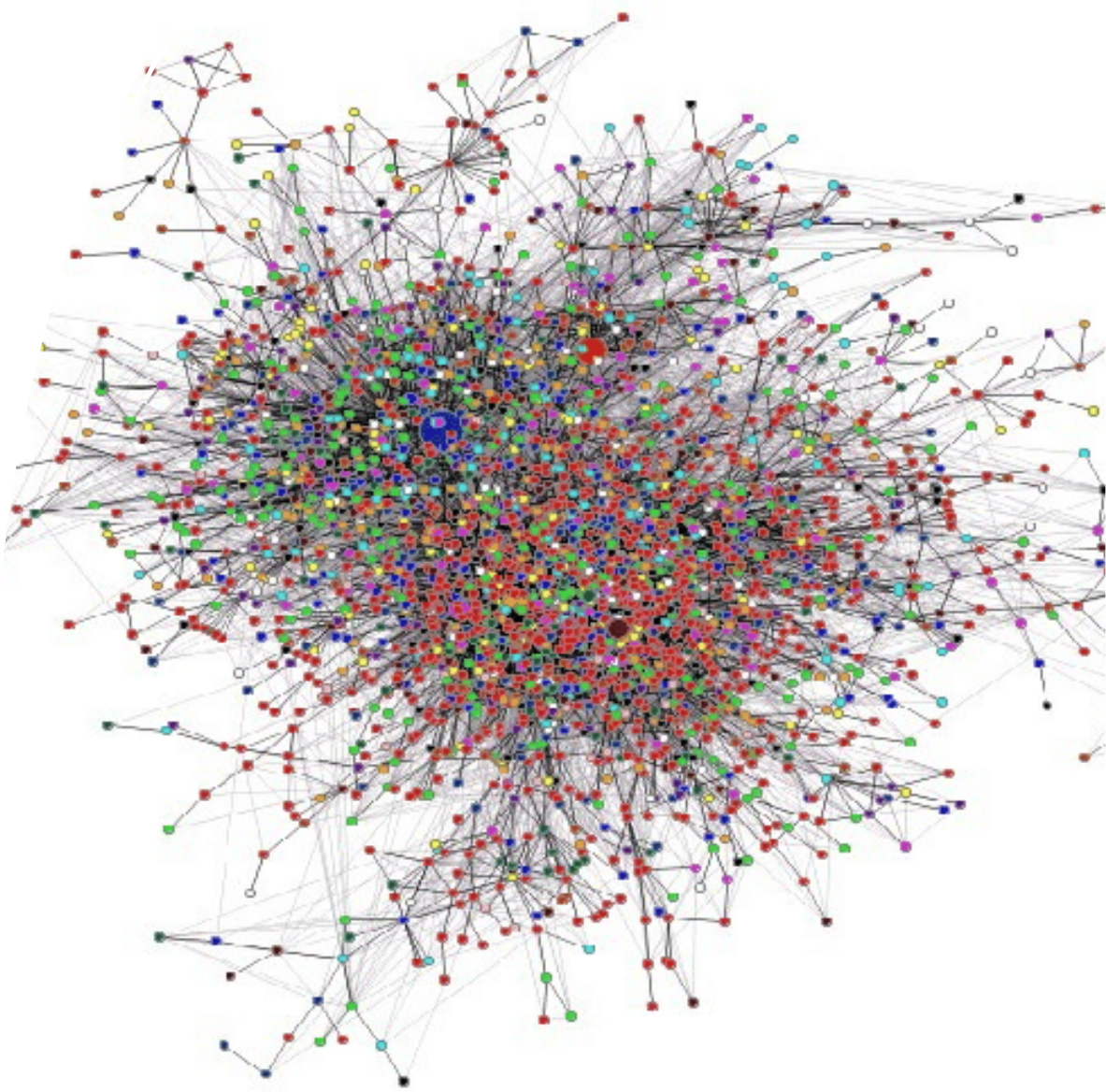


Michelli Brito
@brito_michelli

Padrões de Microservices

O desconhecimento sobre os principais Microservices Patterns é um grande risco na elaboração de Arquiteturas de Microservices.

Base de dados por Microservice diminui o acoplamento, mas é necessário saber lidar corretamente com a sincronia de dados e com consistência eventual por os dados estarem distribuídos, por exemplo.



Michelli Brito
@brito_michelli

Premissas Importantes

O entendimento do negócio juntamente com a experiência e conhecimentos adquiridos, como o de Microservices Patterns, são essenciais para você fazer as melhores escolhas e definir as melhores soluções e assim ser um profissional diferenciado no mercado.

A discussão de Microservices não deve ser apenas sobre tamanho ou complexidade do negócio, mas sim sobre projetar sistemas que respondam facilmente a mudanças e modernizações tecnológicas.

Michelli Brito
@brito_michelli

Obrigada!

Não perca o próximo dia: **Microservices Patterns e Spring Projects**

Michelli Brito
@brito_michelli