



Instituto Infnet

Aula 5 - Arquitetura de Microsserviços e Mobile

Aula 5

- Exemplo Prático;
- Teorema CAP;
- ADR (Architecture Document Record).

Ementa proposta Aula 5

- Leia o white paper do Postman sobre documentação de serviços;
- capítulo 2 de "API Testing And Development With Postman";



Exemplo Prático de Revisão

Sistema de simulador de provas de concurso

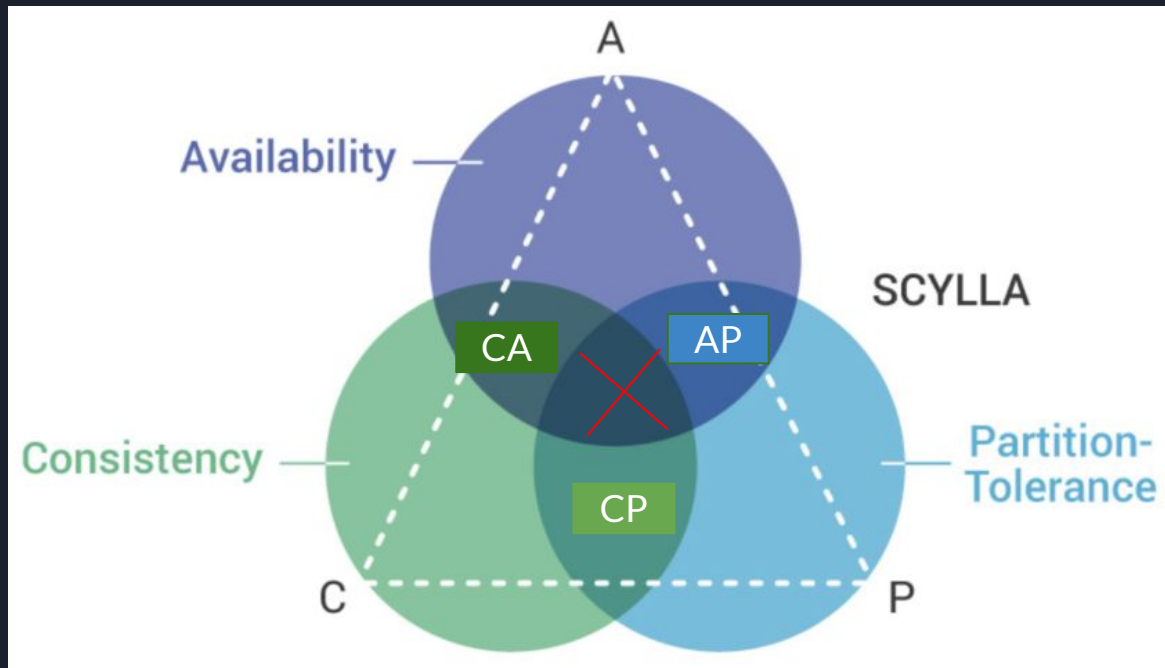
- <https://github.com/engRenanTorres/engenhariaDeConcursos/tree/main>
<https://www.engenhariadeconcursos.com.br/>
- Tecnologias:
React.js -> Nest.js -> MariaDB
- Para o exemplo LocalStack AWS:
- <https://www.localstack.cloud/>



Teorema CAP

*O teorema CAP também é conhecido por Teorema de Brewer.

- Consistência (Consistency);
- Availability (Disponibilidade);
- Partition Tolerance (Tolerância à partição).



Teorema CAP

- Consistência (Consistency):
"A consistência do teorema CAP significa que, independentemente do nó ao qual eles se conectam, todos os clientes veem os mesmos dados de uma vez. Para que a gravação em um nó seja bem-sucedida, ela também deve ser replicada ou encaminhada instantaneamente para todos os outros nós no sistema."

<https://www.scylladb.com/glossary/cap-theorem/>



Consistência CAP \neq Consistência ACID

“Em CAP, consistência significa ter informações que são as mais atualizadas. Consistência em ACID se refere à dureza do banco de dados que o protege de corrupção, apesar da adição de novas transações e referências a diferentes eventos do banco de dados.”

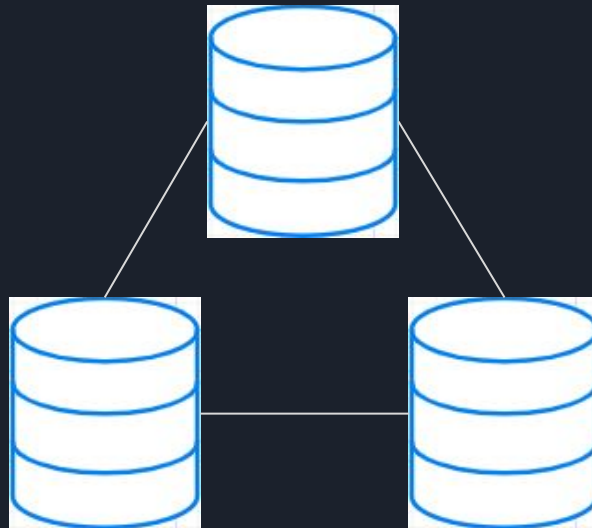
<https://www.scylladb.com/glossary/cap-theorem/>



Teorema CAP

- Availability (Disponibilidade):
“Disponibilidade do teorema CAP significa que mesmo se um ou mais nós estiverem inativos, qualquer cliente que fizer uma solicitação de dados receberá uma resposta. Em outras palavras, quando qualquer solicitação é feita, sem exceção, todos os nós de trabalho em um sistema distribuído retornam uma resposta válida”

<https://www.scylladb.com/glossary/cap-theorem/>



Teorema CAP

- Partition Tolerance (Tolerância à partição):
“Em um sistema distribuído, uma partição é uma interrupção nas comunicações — uma conexão temporariamente atrasada ou perdida entre nós. Tolerância de partição significa que, apesar de qualquer número de interrupções na comunicação entre nós no sistema, o cluster continuará a funcionar”

<https://www.scylladb.com/glossary/cap-theorem/>



Teorema CAP

- Bancos de dados CP:
“Um banco de dados CP oferece consistência e tolerância de partição, mas sacrifica a disponibilidade. O resultado prático é que, quando ocorre uma partição, o sistema deve tornar o nó inconsistente indisponível até que possa resolver a partição. MongoDB e Redis são exemplos de bancos de dados CP.”

<https://www.scylladb.com/glossary/cap-theorem/>



Teorema CAP

- Bancos de dados CP:
ex. MongoDB: “O MongoDB é um sistema de gerenciamento de banco de dados NoSQL popular que é usado para aplicativos de big data em execução em vários locais. O MongoDB resolve partições de rede mantendo a consistência, sacrificando a disponibilidade conforme necessário em caso de falha. Um nó primário recebe todas as operações de gravação no MongoDB. O sistema precisa eleger um novo nó primário se o existente ficar indisponível e, enquanto isso, os clientes não podem fazer nenhuma solicitação de gravação para que os dados permaneçam consistentes.”

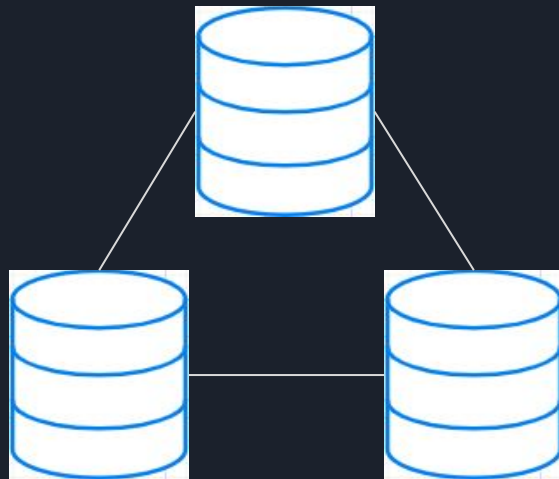
<https://www.scylladb.com/glossary/cap-theorem/>



Teorema CAP

- Bancos de dados AP:
“Um banco de dados AP fornece disponibilidade e tolerância de partição, mas não consistência no caso de uma falha. Todos os nós permanecem disponíveis quando ocorre uma partição, mas alguns podem retornar uma versão mais antiga dos dados. CouchDB, Cassandra e ScyllaDB são exemplos de bancos de dados AP.”

<https://www.scylladb.com/glossary/cap-theorem/>



Teorema CAP

- Bancos de dados AP:
ex. Cassandra: “é um banco de dados NoSQL de código aberto com uma arquitetura peer-to-peer e potencialmente vários pontos de falha. O teorema CAP no Cassandra revela um banco de dados AP: o Cassandra oferece disponibilidade e tolerância à partição, mas não pode fornecer consistência o tempo todo. No entanto, ao reconciliar inconsistências o mais rápido possível e permitir que os clientes gravem em qualquer nó a qualquer momento, o Cassandra fornece **consistência eventual**. As inconsistências são resolvidas rapidamente na maioria dos casos de partições de rede, portanto, a disponibilidade constante e o alto desempenho geralmente valem a troca do teorema CAP do Cassandra”

<https://www.scylladb.com/glossary/cap-theorem/>

Teorema CAP


- Bancos de dados CA:
 - A maioria dos SQL tradicionais
- “...não há bancos de dados NoSQL que você classificaria como CA sob o teorema CAP . Em um banco de dados distribuído , não há como evitar partições do sistema . Portanto, embora o teorema CAP afirmando que um banco de dados distribuído CA seja possível, atualmente não há um verdadeiro sistema de banco de dados distribuído CA. O objetivo moderno da análise do teorema CAP deve ser que os projetistas de sistemas gerem combinações ótimas de consistência e disponibilidade para aplicativos específicos.”

<https://www.scylladb.com/glossary/cap-theorem/>



Teorema CAP aplicado a Microserviços

“...se a capacidade de iterar rapidamente o modelo de dados e escalar horizontalmente é essencial para sua aplicação, mas você pode tolerar consistência eventual (em oposição à consistência estrita), um banco de dados AP como Cassandra ou Apache CouchDB pode atender às suas necessidades e simplificar sua implementação. Por outro lado, se sua aplicação depende fortemente da consistência dos dados, como em uma aplicação de eCommerce ou um serviço de pagamento, você pode optar por um banco de dados relacional como PostgreSQL.”



<https://www.ibm.com/br-pt/topics/cap-theorem>

SQL seguem o ACID

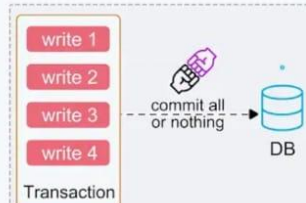
- Banco de dados:
 - Atomicity;
 - Consistency;
 - Isolation;
 - Durability.

What does ACID Mean?

ByteByteGo

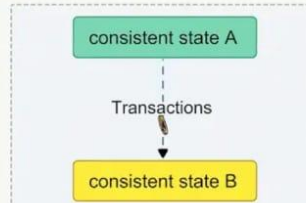
Atomicity

All or nothing



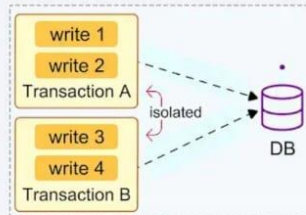
Consistency

Preserving database invariants



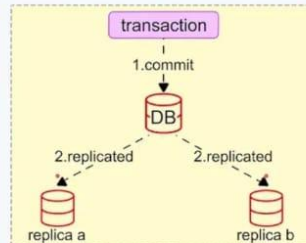
Isolation

Concurrent transactions are isolated from each other



Durability

Data is persisted after transaction is committed even in a system failure



<https://bytebytego.com/>

Quando usar NoSQL

- Escalabilidade Vertical:



ACID Dificulta escalabilidade horizontal:



ACID vs BASE

- “Os bancos de dados ACID priorizam a consistência em detrimento da disponibilidade: toda a transação falhará se ocorrer um erro em qualquer etapa dela. Em contraste, os bancos de dados BASE priorizam a disponibilidade em vez da consistência. Em vez de falhar na transação, os usuários podem acessar dados inconsistentes temporariamente. A consistência de dados é alcançada, mas não imediatamente.”



BASE - Basically Available, Soft State e Eventually Consistent

- Basicamente disponível;
- Estado flexível;
- Leitura eventualmente consistente.



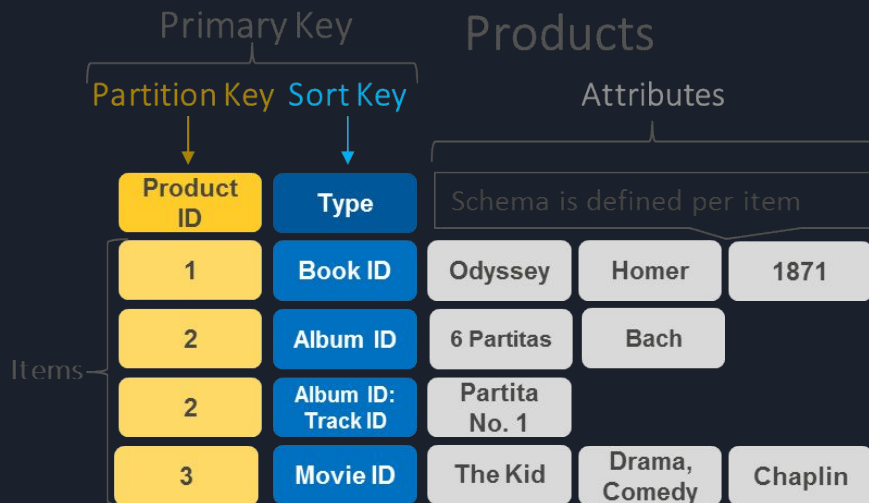
Quando usar NoSQL

- Impedance Mismatch
 - Padrões de dados complexos:

```
{
  "RegistroCompra": {
    "usuario": {
      "nome": "Fulano",
      "cpf": "9817268912",
      "contatos": {
        "endereco": {"rua": "rua x"}
      }
    }
  },
  "usuario": {
    "nome": "Fulano",
    "cnpj": "9817268912",
    "contatos": {
      "endereco": {"rua": "rua x"}
    }
  }
}
```

Tipos de NoSQL

- Bancos de Chave-valor (Key Value Store):
 - Elementos independentes garantem alta performance;
 - Muito utilizados em cache. Ex.: Dynamo, Redis.



<https://aws.amazon.com/pt/nosql/key-value/>

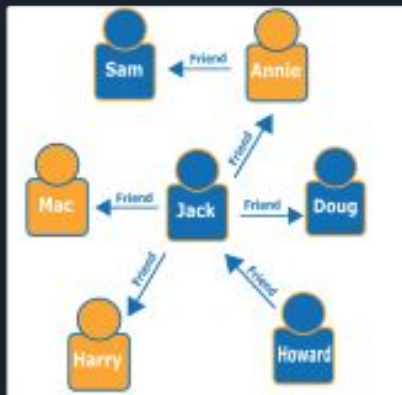
Tipos de NoSQL

- Bancos de Documentos (Document DB):
 - Armazenamento em json(bson), ou xml;
 - Utilizado para armazenar settings, sistemas de vendas e etc.. Ex.: MongoDB.



Tipos de NoSQL

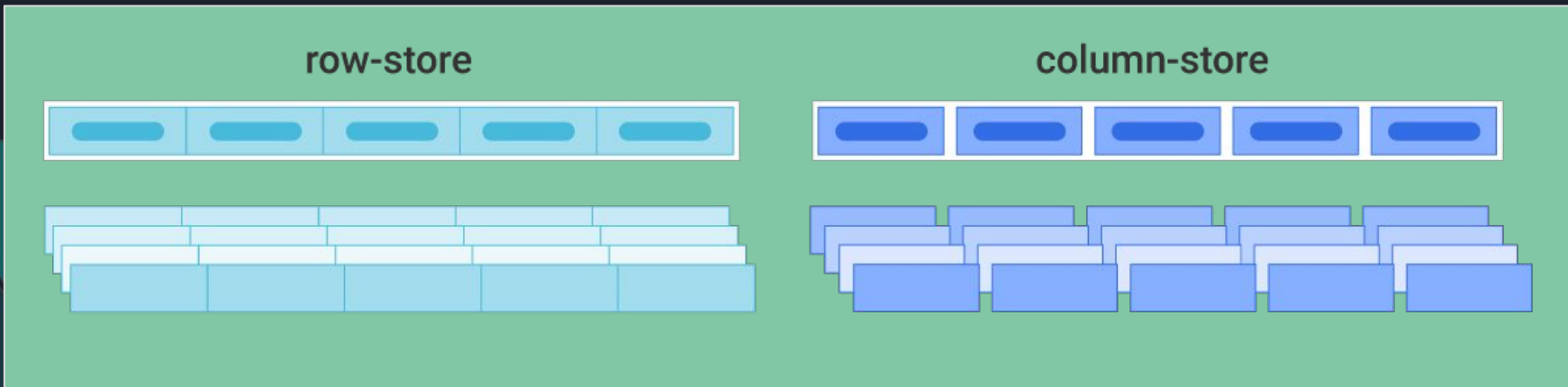
- Banco de dados Grafos (Graph DB):
 - Armazenamento dados em nós(entidade) e edges(relação entre entidades);
 - Utilizado em detecção de fraudes financeiras e sistema de recomendação.
 - Ex.: Neo4j.



<https://aws.amazon.com/pt/nosql/graph/>

Tipos de NoSQL

- Banco de dados de Colunas (Columnar DB):
 - Sistema baseado em armazenamento de colunas, acessa as colunas de modo muito mais eficiente;
 - Utilizado em bigdatas.
 - Ex.: Cassandra, ScyllaDB, Redshift.



<https://www.scylladb.com/glossary/columnar-database/>

Tipos de NoSQL

- Banco de dados Vetorizados (Vector DB):
 - Especializado em guardar matrizes e vetores;
 - Utilizado em Aprendizado de máquina;
 - Ex.: PgVector(PostgreSQL), MongoDB.



Resumo NoSQL x SQL

NoSQL:

- Schema Flexível;
- Escalabilidade Horizontal;
- Modelos de dados especializados;
- Big Data;
- Performance.

SQL:


- ACID;
- Maturidade;
- Queries Complexas;
- Schema fixo/ Organizado;
- Consistência (CAP).



<https://www.youtube.com/watch?v=o8HEfReQ6co&t=8s>

Registros de decisão de Arquitetura -ADR

- Criado por Michael Nygard -
<https://cognitect.com/blog/2011/11/15/documenting-architecture-decisions>
- Uma das formas mais eficazes de documentar decisões;
- Lembre-se: **“Por quê?” é mais importante do que “como?”** em arquitetura de soft.;
- Pequeno arquivo de texto.



Fonte deste tema: Cap. 19 - Fundamentos da Arquitetura de Software (Mark Richards e Neal Ford)

<https://learning.oreilly.com/library/view/fundamentals-of-software/9781492043447/>

Ferramentas para auxílio em ADR

- ADR-tools - [https://github.com/npryce/adr-tools](https://github.com/npryce/adr-tools;);
- Blog com exemplos - Micha
<https://www.hascode.com/managing-architecture-decision-records-with-adr-tools/>



Estrutura básica - ADR

- Título - Breve descrição da decisão;
- Status - (Proposto, Aceito, Substituído);
- Contexto - O que está me forçando a tomar esta decisão?;
- Decisão - A decisão e a justificativa correspondente;
- Consequências - Quais são os impactos desta decisão?;
- Conformidade(Extra) - Como eu garantirei o cumprimento desta decisão?
- Notas(Extra) - Metadata para esta decisão(Autor, etc..)

	ADR Format
	TITLE
	Short description stating the architecture decision
	STATUS
	Proposed, Accepted, Superseded
	CONTEXT
	What is forcing me to make this decision?
	DECISION
	The decision and corresponding justification
	CONSEQUENCES
	What is the impact of this decision?
	COMPLIANCE
	How will I ensure compliance with this decision?
	NOTES
	Metadata for this decision (author, etc.)

Estrutura básica - ADR

- Título -
O título de um ADR geralmente é numerado sequencialmente. Ele deve ser uma frase curta, mas concisa e sem ambiguidades e que descreva a decisão de arquitetura.
Ex.:
"42. Uso de mensagens assíncronas entre os Order and Payment Services.";
- Status - (Proposto, Aceita, Substituído) Status proposto significa que a decisão deve ser aprovada por um tomador de decisão de nível superior ou algum tipo de órgão de governança arquitetônica (como um conselho de revisão de arquitetura)
Ex.:
"ADR 42. Uso de mensagens assíncronas entre serviços de pedidos e pagamentos
Status: Substituído por 68"

"ADR 68. Uso de REST entre serviços de pedidos e pagamentos
Status: Aceito, substitui 42";

Estrutura básica - ADR

- Contexto - O que está me forçando a tomar esta decisão?
Esta seção do ADR permite que o arquiteto descreva a situação ou problema específico e elabore concisamente as alternativas possíveis. Em caso de muitas alternativas, pode-se criar uma seção chamada "alternativas";
Ex.: "O serviço de pedidos deve passar informações para o serviço de pagamento para pagar por um pedido que está sendo feito no momento. Isso pode ser feito usando REST ou mensagens assíncronas."
- Decisão - A decisão e a justificativa correspondente
A seção Decisão do ADR contém a decisão de arquitetura, juntamente com uma justificativa completa para a decisão;
Ex.: " usaremos mensagens assíncronas entre serviços"

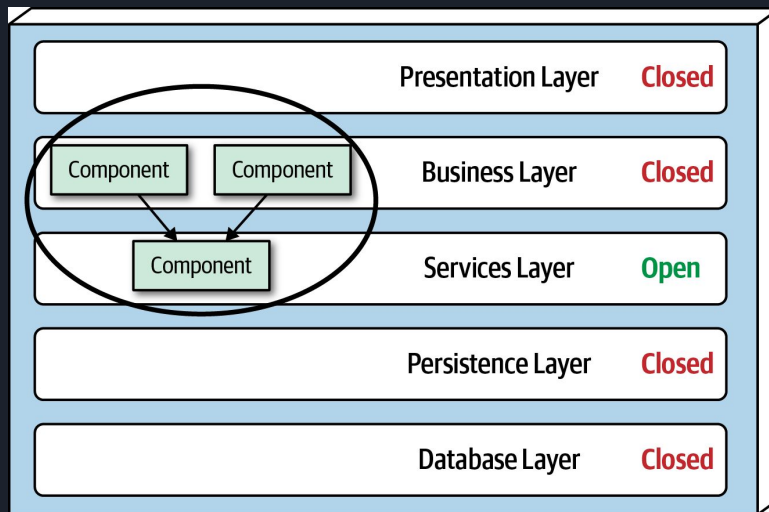
Estrutura básica - ADR

- Consequências- Quais são os impactos desta decisão?
documentar a análise de trade-off associada à decisão de arquitetura. Esses trade-offs podem ser baseados em custo ou trade-offs contra outras características de arquitetura;
- Conformidade(Extra) - Como eu garantirei o cumprimento desta decisão?
A seção Compliance força o arquiteto a pensar sobre como a decisão de arquitetura será medida e governada de uma perspectiva de conformidade.
Ex.: Uso de libs como ArchUnit em Java ou NetArchTest em C# para garantir arquitetura em emcamadas:

```
@Test
public void shared_services_should_reside_in_services_layer() {
    classes().that().areAnnotatedWith(SharedService.class)
        .should().resideInAPackage("..services..")
        .because("All shared services classes used by business " +
            "objects in the business layer should reside in the services " +
            "layer to isolate and contain shared logic")
        .check(myClasses);
}
```

Estrutura básica - ADR

- Conformidade(Extra) - Como eu garantirei o cumprimento desta decisão?
A seção Compliance força o arquiteto a pensar sobre como a decisão de arquitetura será medida e governada de uma perspectiva de conformidade.
Ex.: Uso de libs como ArchUnit em Java ou NetArchTest em C# para garantir arquitetura em emcamadas:



Estrutura básica - ADR

- Notas(Extra) - Metadata para esta decisão(Autor, etc..)

Ex.:

Autor original;

Data de aprovação;

Aprovado por;

Data de substituição;

Data da última modificação;

Modificado por;

Última modificação.



Resumo dos Temas vistos





Instituto Infnet

Aula 5 - Arquitetura de Microsserviços e Mobile