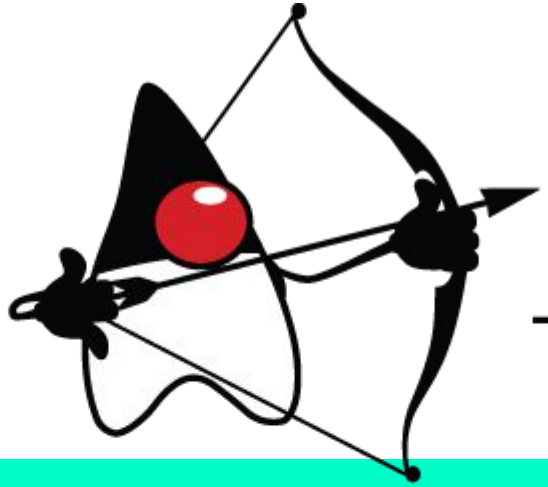
Conhecendo o

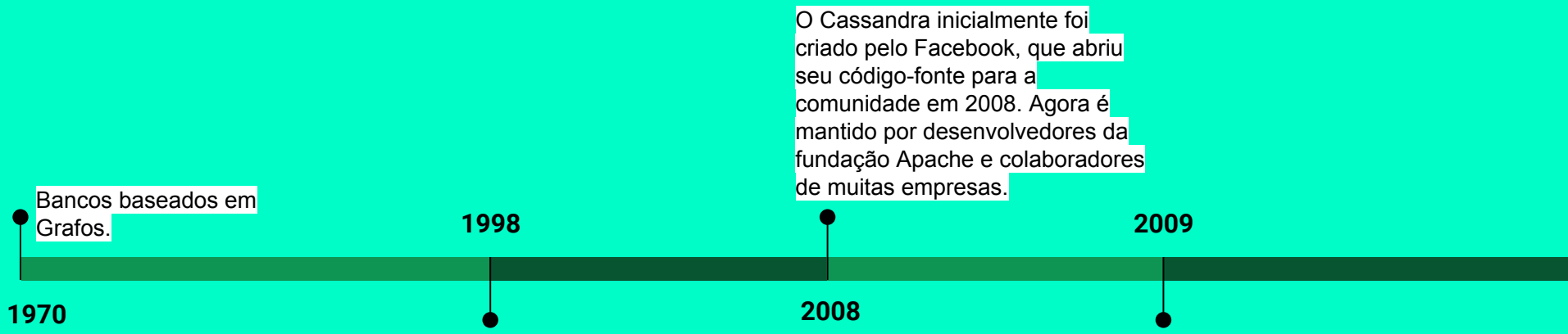


JNOSQL

~$WHOAMI

- Caiçara, 28 anos, de Praia Grande
- Trabalho com tecnologia desde 2008
- Programadora Java desde 2014
- Arquiteta de Soluções Digitais
- Palestrante e Mentora na horas vagas
- Entusiasta em Agilidade
- Ativista da cultura do compartilhamento

NoSQL


Bancos baseados em Grafos.
1970
1998
O termo NoSQL foi primeiramente utilizado em 1998 como o nome de um banco de dados relacional de código aberto que não possuía uma interface SQL. Seu autor, Carlo Strozzi.
2008
O Cassandra inicialmente foi criado pelo Facebook, que abriu seu código-fonte para a comunidade em 2008. Agora é mantido por desenvolvedores da fundação Apache e colaboradores de muitas empresas.
2009
O termo **NoSQL** foi re-introduzido no início de 2009 por um funcionário do Rackspace, Eric Evans, quando Johan Oskarsson da Last.fm queria organizar um evento para discutir bancos de dados open source distribuídos.

Características

Ausência de Esquema ou esquema flexivel

Suporte a replicação de dados

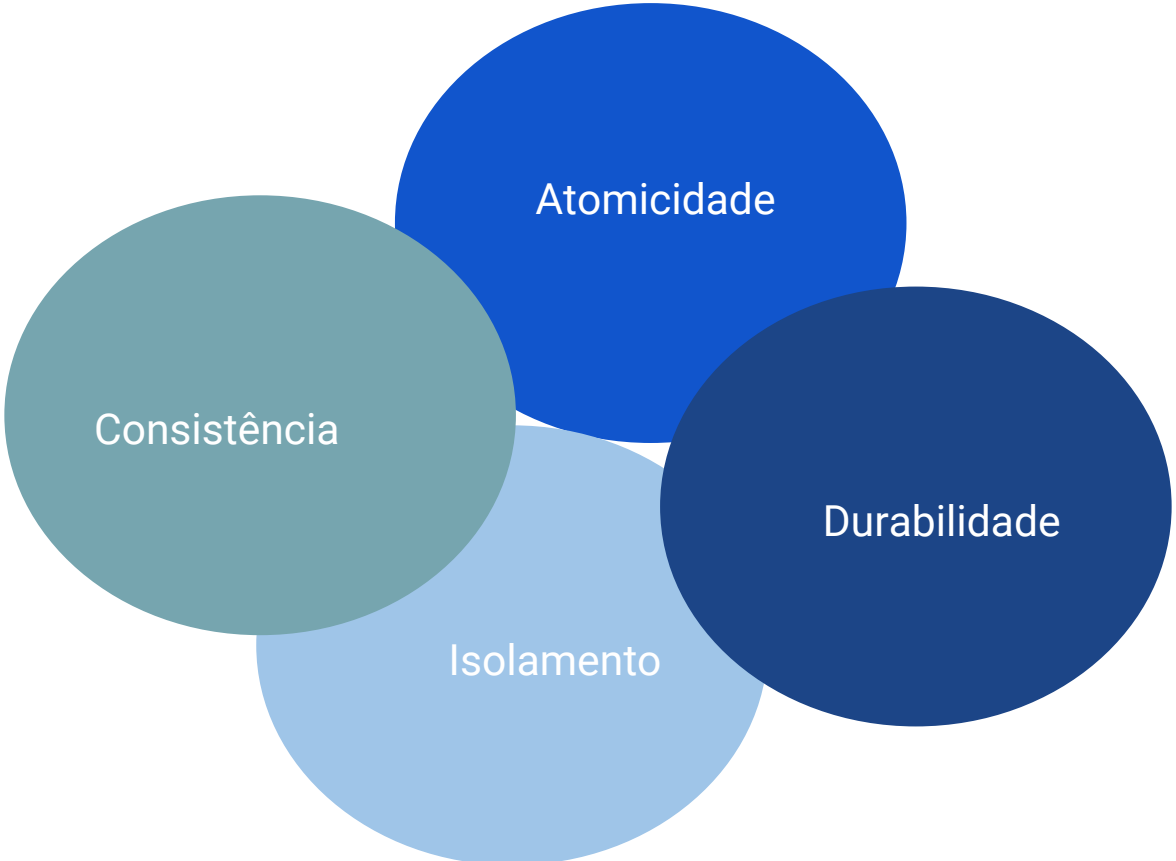
Escalabilidade horizontal

Nem sempre consistente

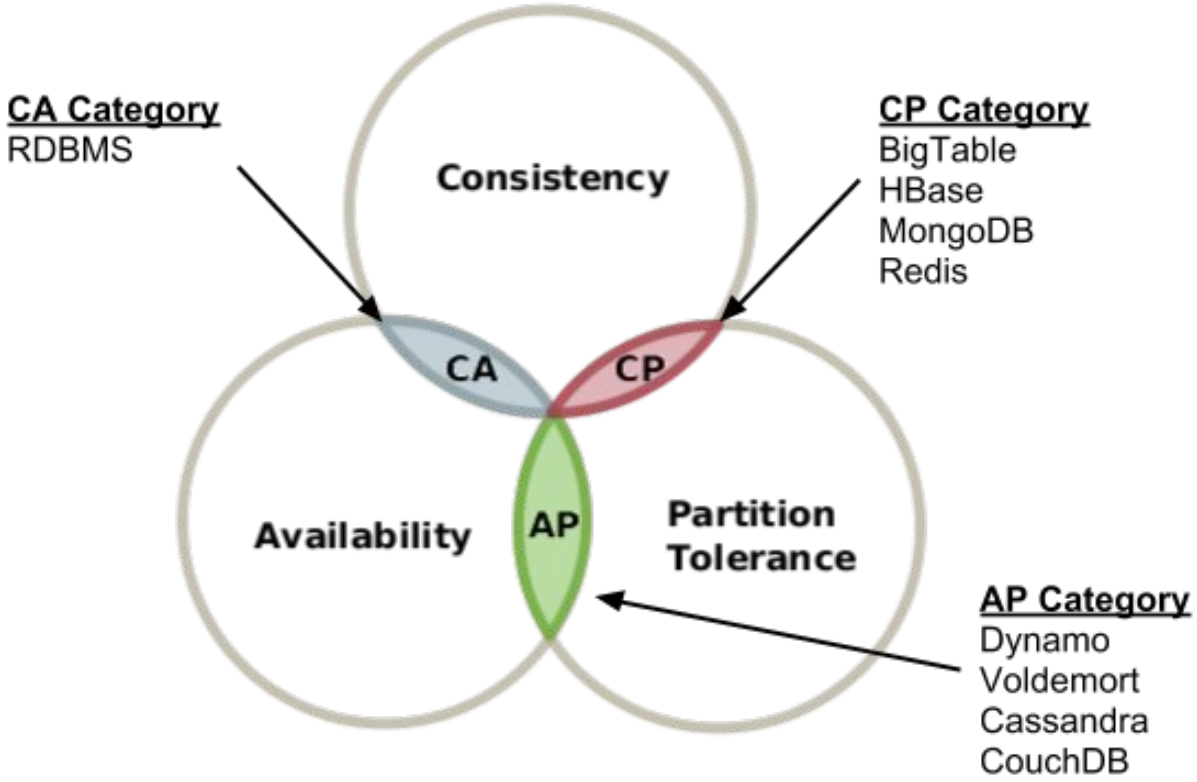
NOSQL -> BASE


Basicamente
Disponível
Estado Leve
Consistência
Eventual

RELACIONAL -> ACID

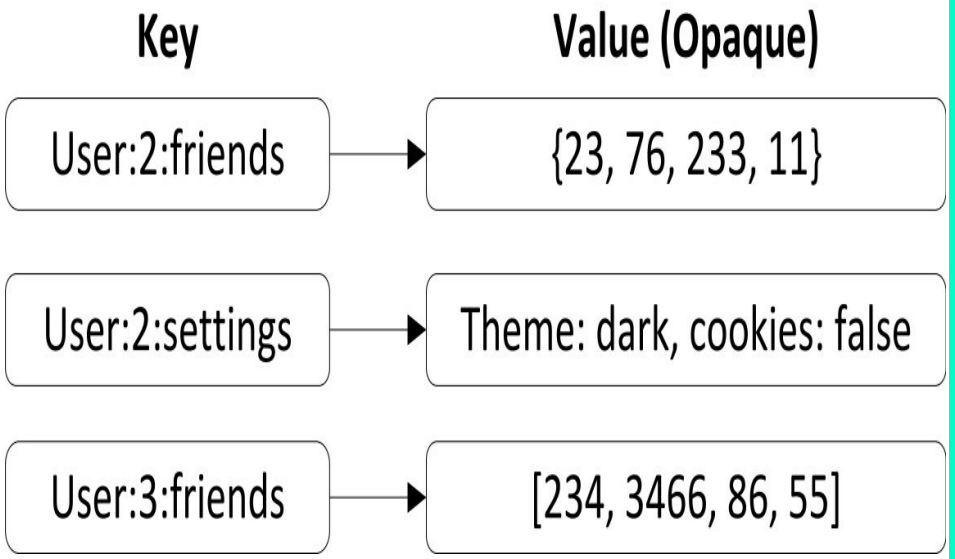


Teorema CAP



Consistência	*Disponibilidade (Availability)*	*Partição tolerante a falhas*
Cada leitura recebe a escrita mais recente ou um erro	Cada pedido recebe uma resposta (sem erro) - sem garantia de que contém a escrita mais recente	O sistema continua a funcionar apesar de um número arbitrário de mensagens serem descartadas (ou atrasadas) pela rede entre nós

Chave - Valor



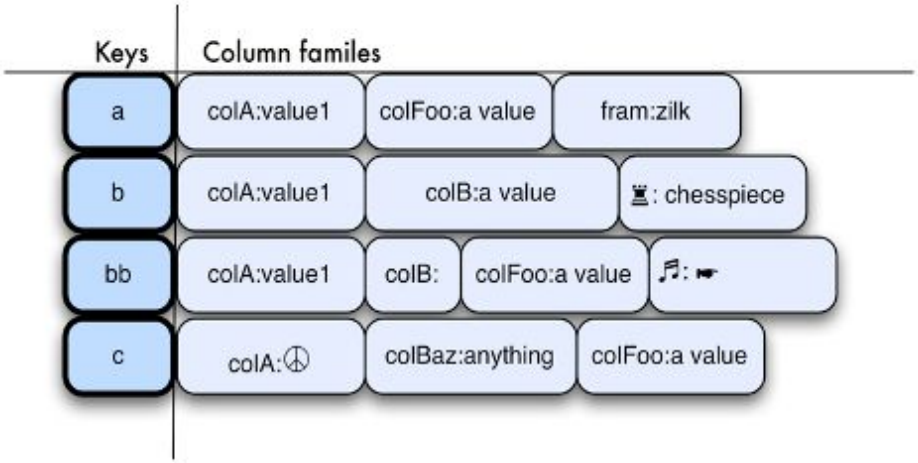
`Redis`

`Amazon DynamoDB`

`AmazonS3`

`Helze`

Família de Colunas



HBASE

Cassandra

DynamoDb

SimpleDb

```
{
  "id": 55,
  "Pais": "Brasil",
  "Regiao":  "América do Sul",
  "Populacao": 201032714,
  "PrincipaisCidades": [
    {
      "NomeCidade": "São Paulo",
      "Populacao": 1182876,
    },
    {
      "NomeCidade": "Rio de Janeiro",
      "Populacao": 6323037,
    }
  ]
}
```

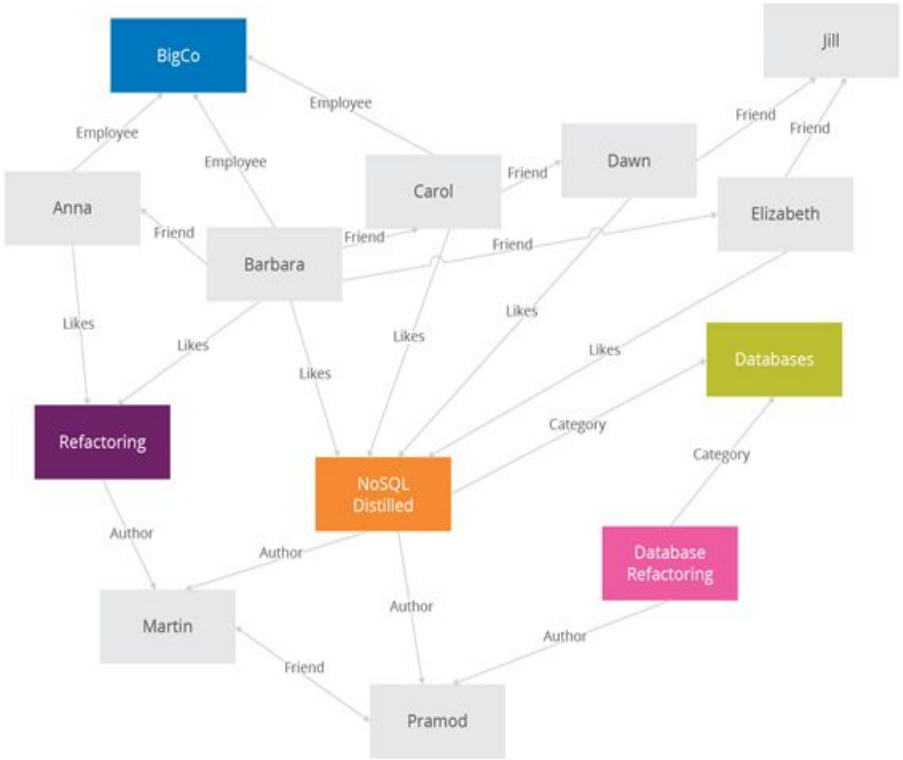
Documento

MongoDB

ApacheCouchDB

DynamoDb

SimpleDb



Grafo

Neo4j

InfoGrid

Sones

HyperGraphDB

Multimodel

OrientDB

CouchBase

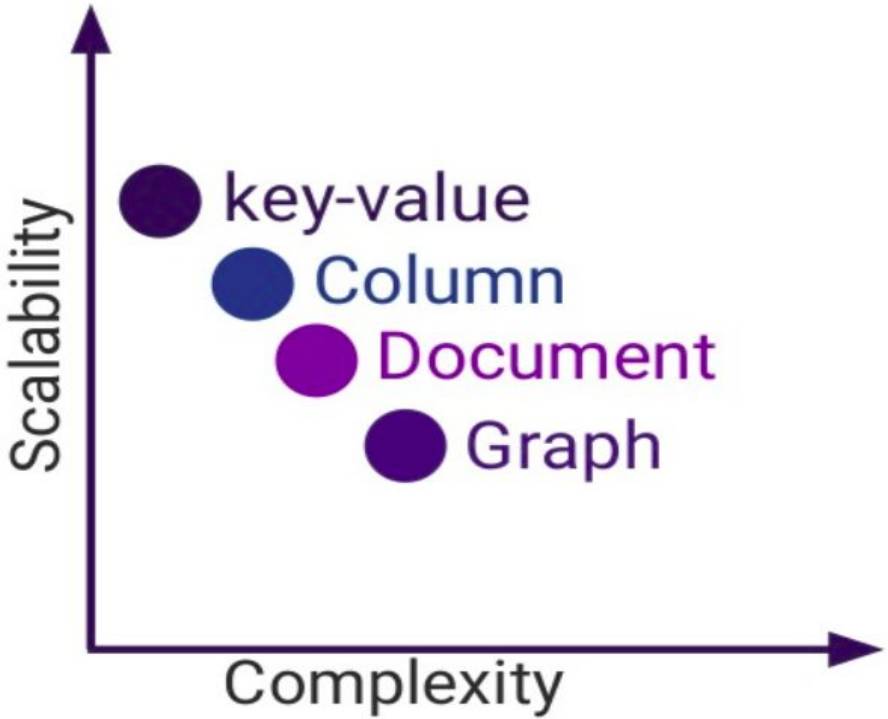
ArangoDB

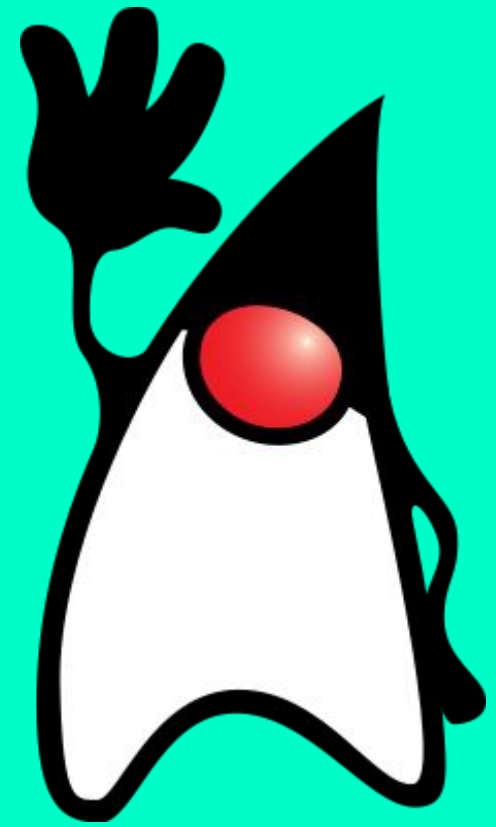
Elasticsearch

SQL	KEY-VALUE	COLUMN	DOCUMENTS	GRAPH
Table	Bucket	Column family	Collection	
Row	Key/value pair	column	Documents	Vertex
Column		Key/value pair	Key/value pair	Vertex and Edge property
Relationship			Link	Edge

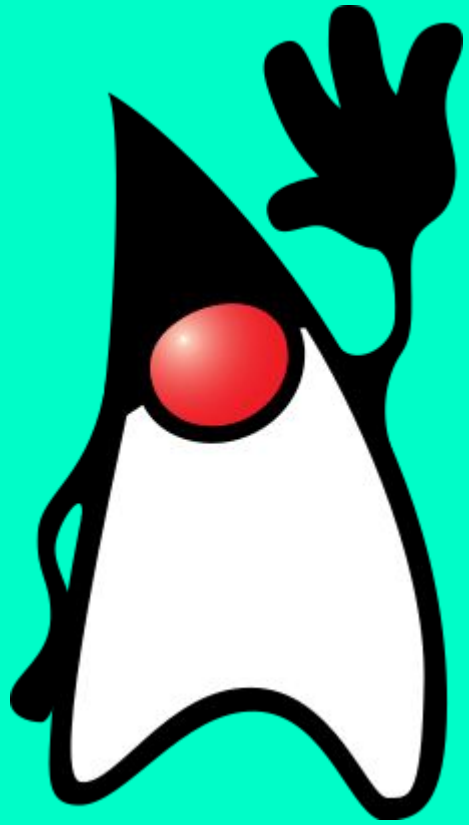
SQL > Nosql

Escalabilidade vs complexidade



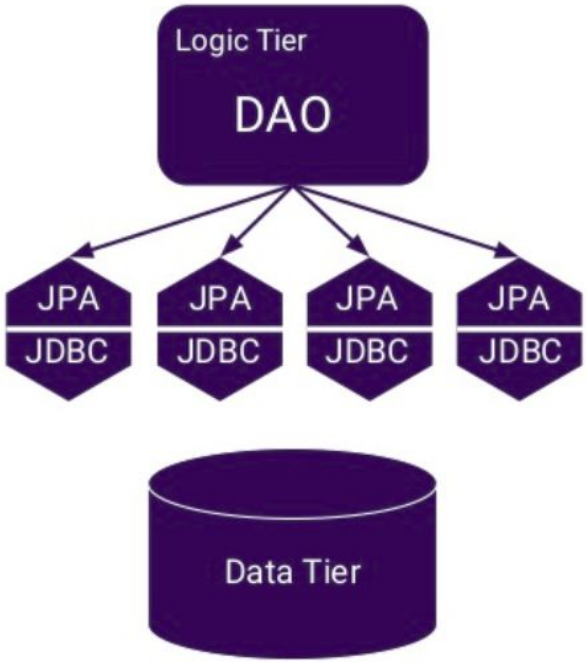


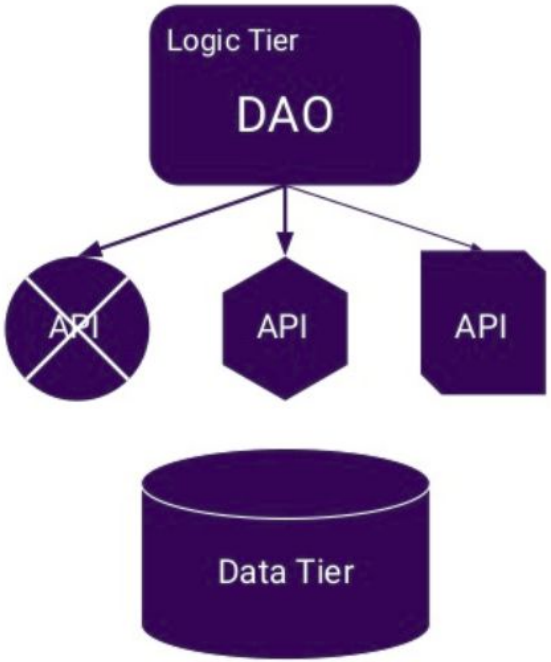
JNOSQL



Document Database	Graph Databases
 Couchbase MarkLogic mongoDB	 InfiniteGraph The Distributed Graph Database
Wide Column Stores	**Key-Value Databases**
   	  HYPERTABLE INC   Amazon SimpleDB

SQL vs Nosql


Logic Tier
DAO
JPA
JDBC
JPA
JDBC
JPA
JDBC
JPA
JDBC
Data Tier


Logic Tier
DAO
API
API
API
Data Tier

O projeto JNoSQL é um projeto que visa a criação de ferramentas para o desenvolvimento com NoSQL, seu foco é padronizar.

Vantagens

O QUE VOCÊ DEVE SABER

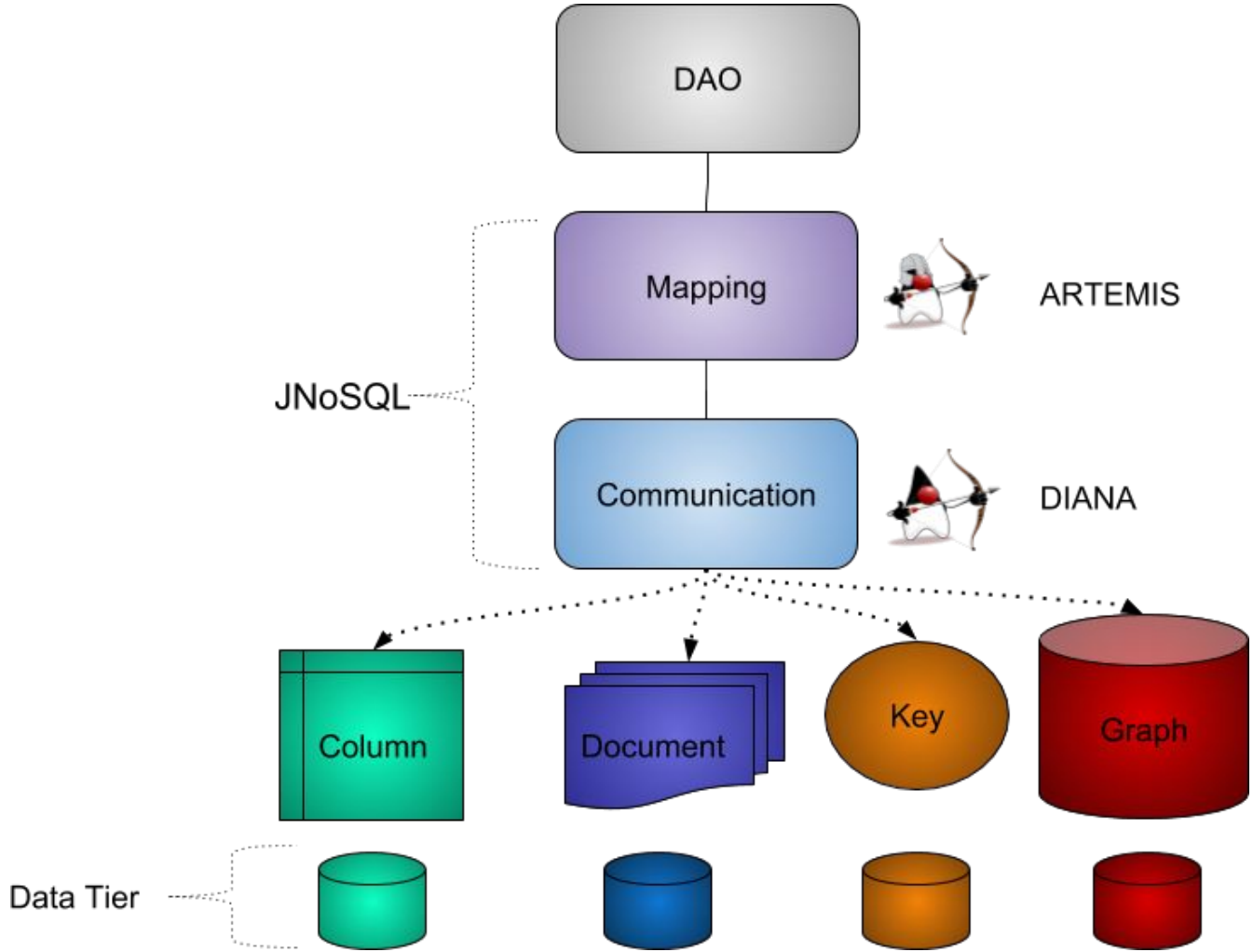
- Primeira versão foi disponibilizada em 15-03-2017
- Liderado pelo brasileiro Otávio Santana
- Projeto realizado por diversos brasileiros
- Podemos contribuir também

Diana

- Responsável por realizar a comunicação entre a comunicação e o Banco de Dados
- Dividida em 4 partes, sendo cada uma para um tipo de banco de dados Nosql
- Semelhante ao JDBC do mundo relacional

Artemis

- Semelhante a um ORM, abstrai e facilita a implementação para o desenvolvedor
- Utiliza anotações
- Permite integração com outras ferramentas como o Bean Validation


DAO
Mapping
ARTEMIS
JNoSQL
Communication
DIANA
Column
Document
Key
Graph
Data Tier

Vantagens dessa divisão

Divisão de Problemas

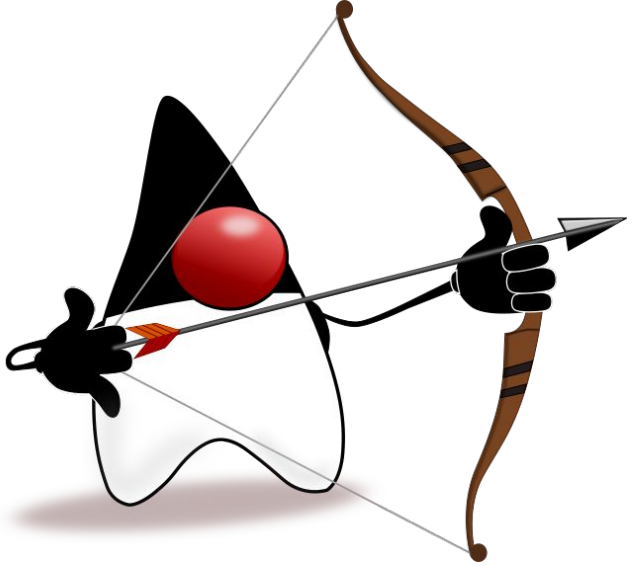
(Assim, os bancos de dados darão atenção apenas a camada de comunicação enquanto desenvolvedores de framework darão atenção numa camada superior.

Facilidade na implementação

Uma vez um novo banco de dados interessado em implementar a API do JnoSQL será necessário apenas implementar a API de comunicação não se preocupando com as outras camadas.

Facilidade na componentização

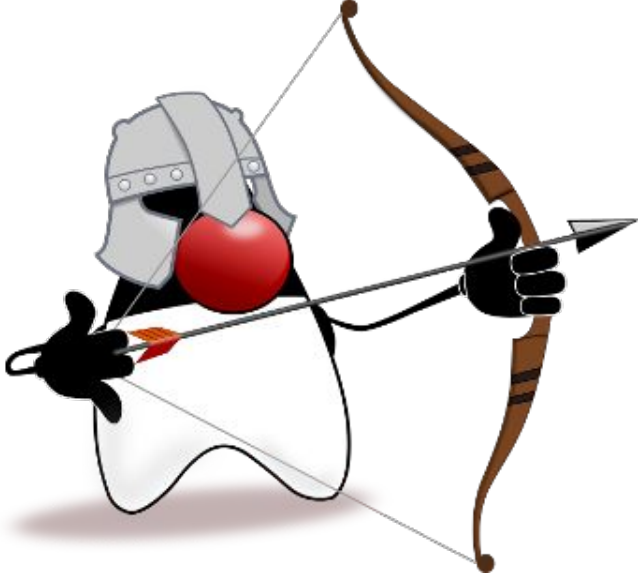
Com essa estratégia será possível trocar um dos dois componentes sem que necessariamente exista impacto no outro lado.



- diana-core
- diana-key-value
- diana-column
- diana-document

Apache ThinkerPop

```
<dependency>
    <groupId>org.jnosql.diana</groupId>
    <artifactId>diana-key-value</artifactId>
    <version>0.0.5</version>
</dependency>
<dependency>
    <groupId>org.jnosql.diana</groupId>
    <artifactId>redis-driver</artifactId>
    <version>0.0.6</version>
</dependency>
```



- artemis-core
- artemis-key-value
- artemis-column
- artemis-document
- artemis-configuration

```
@Entity

@Column

@Id

@Embeddable

@MappedSuperClass

@Convert

@Database
```

Posso contribuir com o projeto?

- Documentação
- Revisar a documentação já existente
- Feedback na API
- Encontrar bugs na implementação
- Implementar novos drivers
- Criar exemplos
- Ajudar na tradução do material para o seu idioma
- Realizar a palestra sobre esse projeto no seu JUG.

GitHub

LinkedIn

psanrosa13@gmail.com

Referências

- https://imasters.com.br/banco-de-dados/bancos-de-dados-nosql-uma-visao-geral
- https://otaviojava.wordpress.com/2017/01/12/jnosql_q__a/
- https://jnosql.gitbook.io/jnosql-book
- https://github.com/eclipse/jnosql-artemis