

# Sistemas de Apoio às Decisões

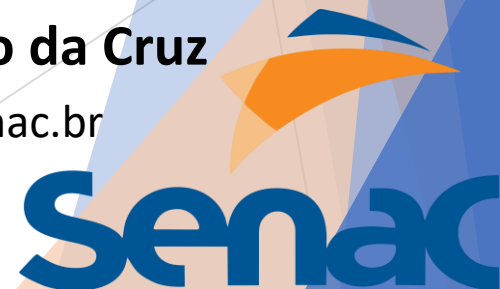
Aula 05 – Introdução à Arquitetura de Dados II

---

**Prof. Esp. Guilherme Jorge Aragão da Cruz**

 [guilherme.jacruz@sp.senac.br](mailto:guilherme.jacruz@sp.senac.br)

 [linkedin.com/in/guijac](https://linkedin.com/in/guijac)



# Roteiro

---

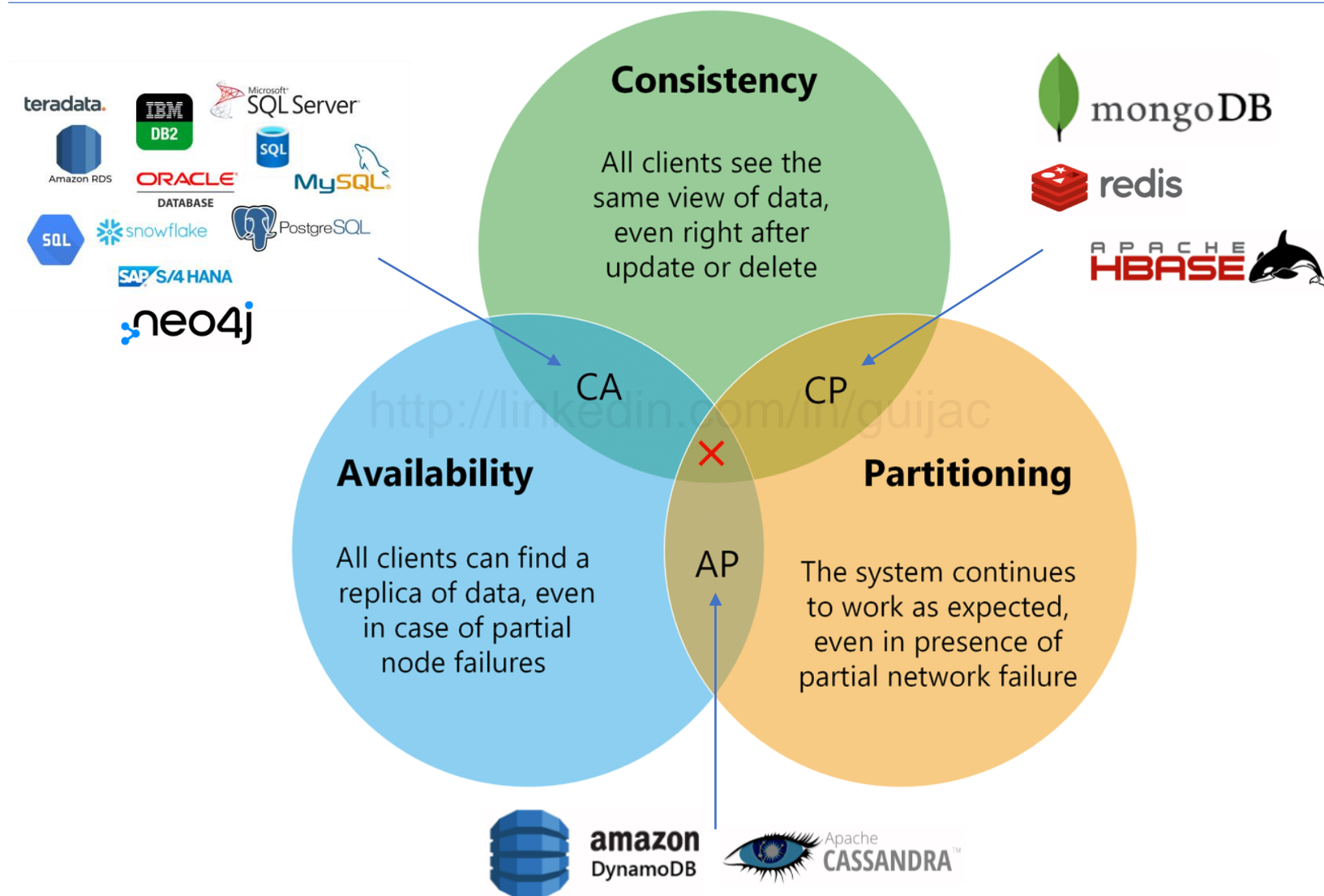
- Teorema CAP;
- 3 Vs;
- Escolhendo um Banco de Dados;
- Arquitetura de Software/Aplicações
  - Definição;
  - Importância.
- Estilos e Padrões Arquiteturais e Padrões de Projeto (*Design Patterns*);
- Padrões Arquiteturais de Aplicações
  - Arquitetura em Camadas;
  - Arquitetura *Model-View-Controller* (MVC);
  - Spring MVC.
- Padrões Arquiteturais em Dados
  - Arquiteturas Centralizadas em Dados;
  - Arquiteturas de Fluxo de Dados.
- Laboratório;
- Referências Bibliográficas.

# Teorema CAP

---

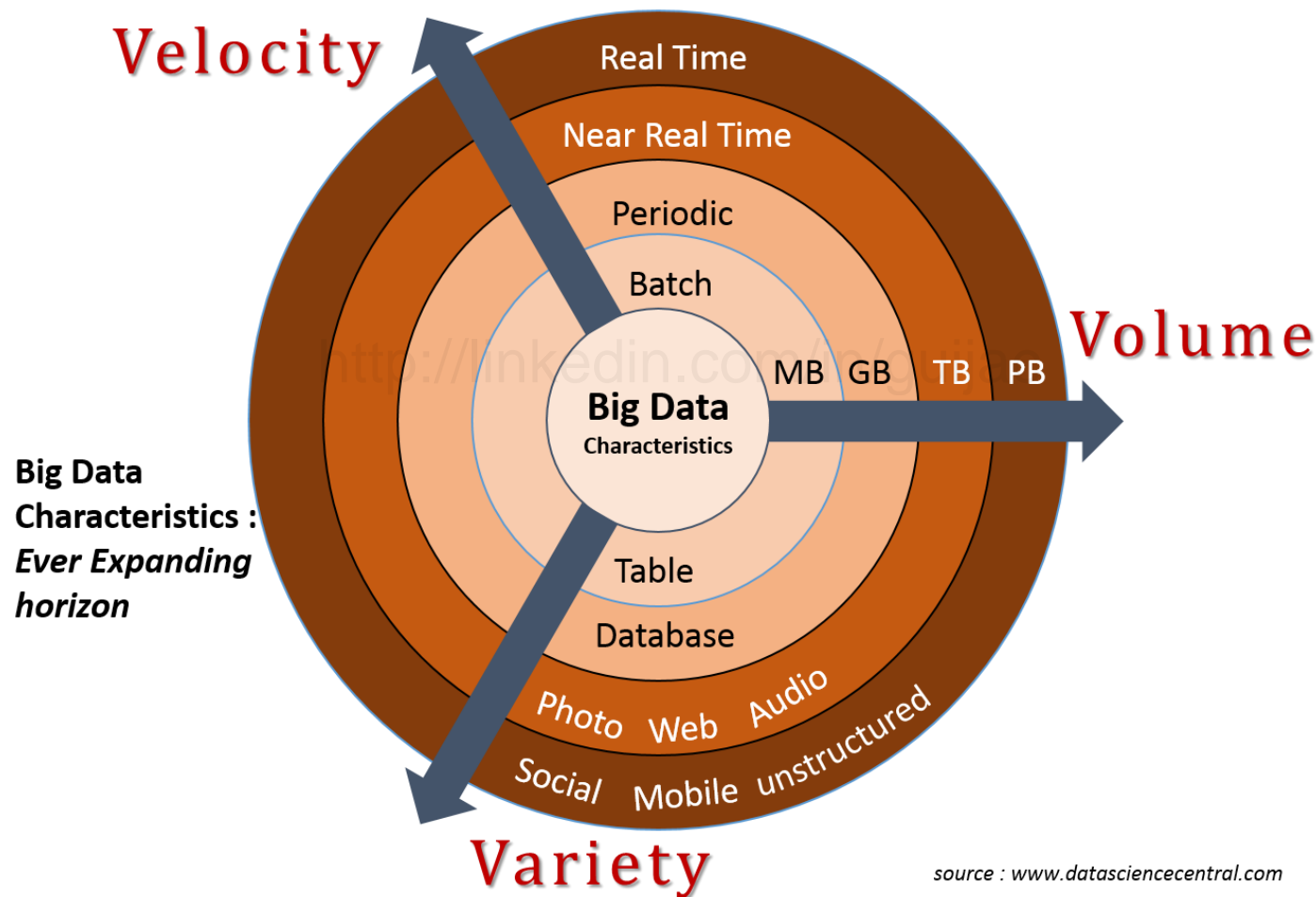
- Também chamado de Teorema de Brewer, afirma que um sistema distribuído, composto por vários nós armazenando dados, não pode fornecer simultaneamente mais de duas das três garantias a seguir:
  - **Consistência:** toda solicitação de leitura recebe a gravação mais recente ou um erro quando a consistência não pode ser garantida;
  - **Disponibilidade:** cada solicitação recebe uma resposta sem erro, mesmo quando os nós estão inativos ou indisponíveis;
  - **Tolerância de partição:** o sistema continua operando apesar da perda de um número arbitrário de mensagens entre os nós.
- **Importante:** na maior parte dos casos, sistemas distribuídos precisam tolerar falhas, logo a escolha deve ser mais direcionada para consistência ou disponibilidade.

# Teorema CAP



Fonte: Adaptado de [Understanding CAP theorem. What is the CAP theorem? | by Nader Medhat](#)

# 3 Vs



Fonte: [Big Data in Science & Problem Solving - Soil Carbon Information](http://www.datasciencecentral.com)

DB-Engines Ranking

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the [method](#) of calculating the scores.



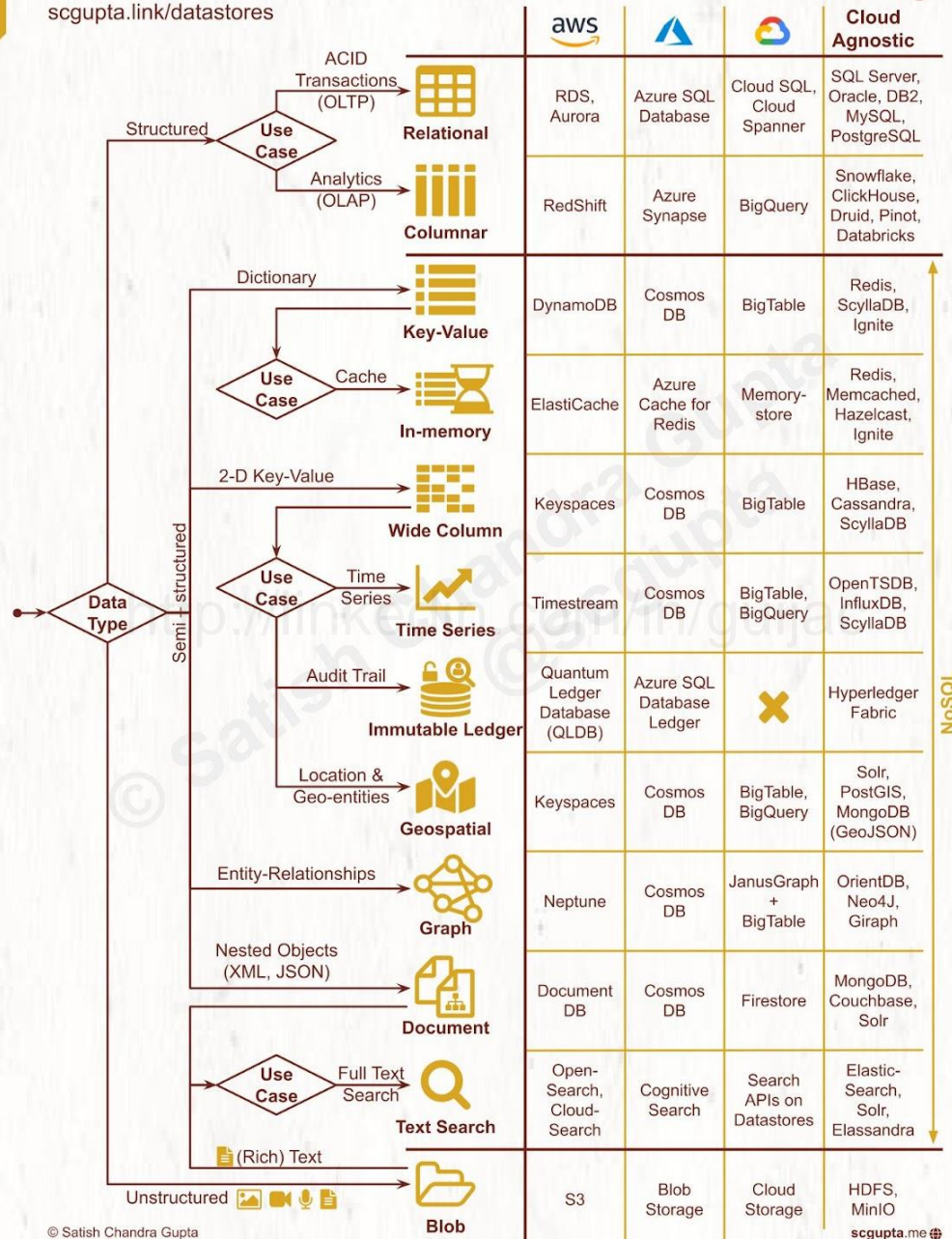
417 systems in ranking, January 2024

Rank			DBMS	Database Model	Score		
Jan 2024	Dec 2023	Jan 2023			Jan 2024	Dec 2023	Jan 2023
1.	1.	1.	Oracle +	Relational, Multi-model ⓘ	1247.49	-9.92	+2.33
2.	2.	2.	MySQL +	Relational, Multi-model ⓘ	1123.46	-3.18	-88.50
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model ⓘ	876.60	-27.23	-42.79
4.	4.	4.	PostgreSQL +	Relational, Multi-model ⓘ	648.96	-1.94	+34.11
5.	5.	5.	MongoDB +	Document, Multi-model ⓘ	417.48	-1.67	-37.70
6.	6.	6.	Redis +	Key-value, Multi-model ⓘ	159.38	+1.03	-18.17
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model ⓘ	136.07	-1.68	-5.09
8.	8.	↓ 7.	IBM Db2	Relational, Multi-model ⓘ	132.41	-2.19	-11.16
9.	↑ 10.	↑ 11.	Snowflake +	Relational	125.92	+6.04	+8.66
10.	↓ 9.	↓ 9.	Microsoft Access	Relational	117.67	-4.08	-15.69
11.	11.	↓ 10.	SQLite +	Relational	115.20	-2.75	-16.29
12.	12.	12.	Cassandra +	Wide column, Multi-model ⓘ	111.04	-1.16	-5.27
13.	13.	13.	MariaDB +	Relational, Multi-model ⓘ	99.23	-1.19	-0.12
14.	14.	14.	Splunk	Search engine	92.72	-3.57	+4.32
15.	15.	↑ 16.	Microsoft Azure SQL Database	Relational, Multi-model ⓘ	81.07	-1.96	+0.70
16.	16.	↓ 15.	Amazon DynamoDB +	Multi-model ⓘ	80.94	-1.47	-0.61
17.	17.	↑ 19.	Databricks +	Multi-model ⓘ	80.53	+0.22	+19.71
18.	18.	↓ 17.	Hive	Relational	66.96	-2.45	-7.39
19.	19.	↑ 21.	Google BigQuery +	Relational	63.48	+1.31	+9.05
20.	20.	↓ 18.	Teradata	Relational, Multi-model ⓘ	53.18	-2.51	-12.25

# Escolhendo um Banco de Dados

## SQL vs. NoSQL: Cheatsheet for AWS, Azure, and Google Cloud

scgupta.link/datastores



© Satish Chandra Gupta  
CC BY-NC-ND 4.0 International License  
creativecommons.org/licenses/by-nc-nd/4.0/

scgupta.me  
twitter.com/scgupta  
linkedin.com/in/scgupta

Fonte: [Understanding Database Types - by Alex Xu \(bytebytego.com\)](https://bytebytego.com/)

# Arquitetura de Software/Aplicações

## Definição

---

<http://linkedin.com/in/guijac>



# Arquitetura de Software/Aplicações

## Definição

---

<http://linkedin.com/in/guijac>



Fonte: [Pedro Peregrina no X / X \(twitter.com\)](https://twitter.com/pedroperegrinaa)

# Arquitetura de Software/Aplicações

## Definição

“A arquitetura de um sistema constitui uma definição abrangente que descreve sua **forma e estrutura** — seus componentes e como eles se integram.”

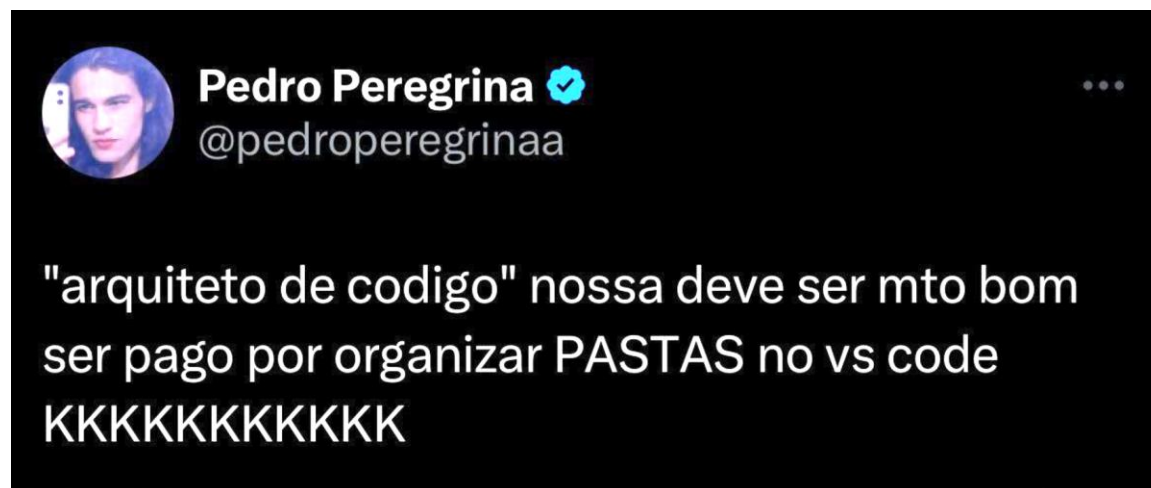
JERROLD GROCHOW (PRESSMANN, 2021)

“A arquitetura de software de um programa ou sistema computacional é a estrutura ou estruturas do sistema, que abrange os **componentes** de software, as **propriedades** externamente visíveis desses componentes e as **relações** entre eles.”

BASS, CLEMENTS E KAZMAN (2013)

“Case-se depressa com sua arquitetura, arrependa-se quando quiser.”

BARRY BOEHM (PRESSMANN, 2021)



Fonte: [Pedro Peregrina no X / X \(twitter.com\)](https://twitter.com/pedroperegrinaa)

# Arquitetura de Software/Aplicações

## Importância

---

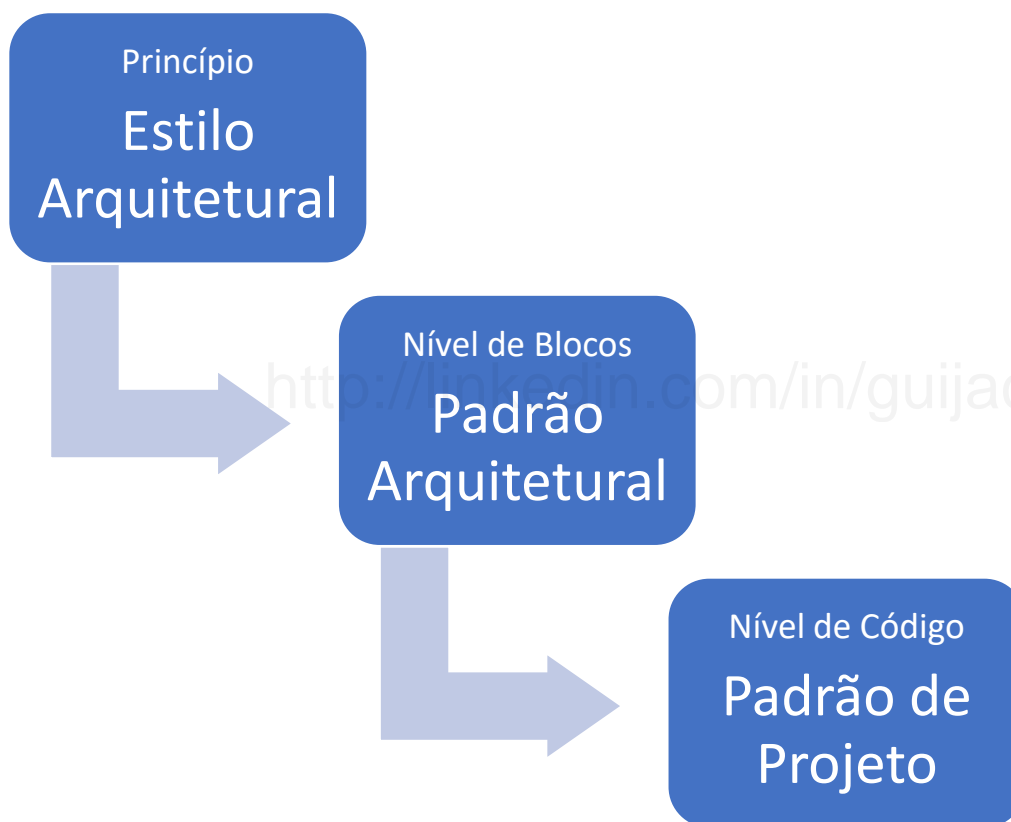
“As representações da arquitetura de software são um **facilitador** para a **comunicação** entre todas as partes interessadas no desenvolvimento de um sistema computacional;

A arquitetura **evidencia decisões** de projeto iniciais que terão profundo impacto em todo o trabalho de engenharia de software que vem a seguir e, tão importante quanto, no sucesso final do sistema como uma entidade operacional;

Constitui um modelo **relativamente pequeno e intelectualmente compreensível** de como o sistema é estruturado e como seus componentes trabalham em conjunto.”

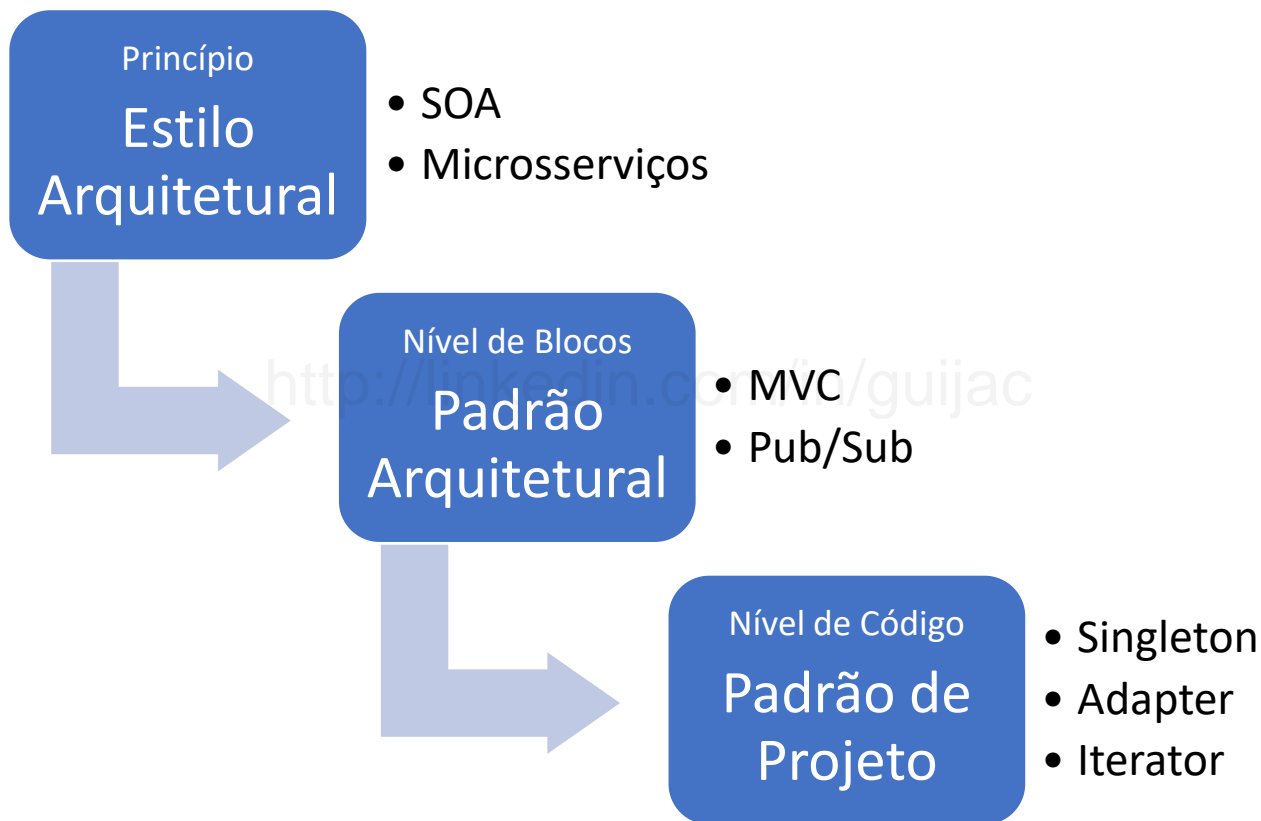
BASS, CLEMENTS E KAZMAN (2013)

# Estilos e Padrões Arquiteturais e Padrões de Projeto (*Design Patterns*)



Fonte: Adaptado de [What is the difference between Design Pattern Vs Architecture Pattern Vs Architecture Style? - Prateek's Blog \(rprateek.com\)](https://prateek.com/what-is-the-difference-between-design-pattern-vs-architecture-pattern-vs-architecture-style/)

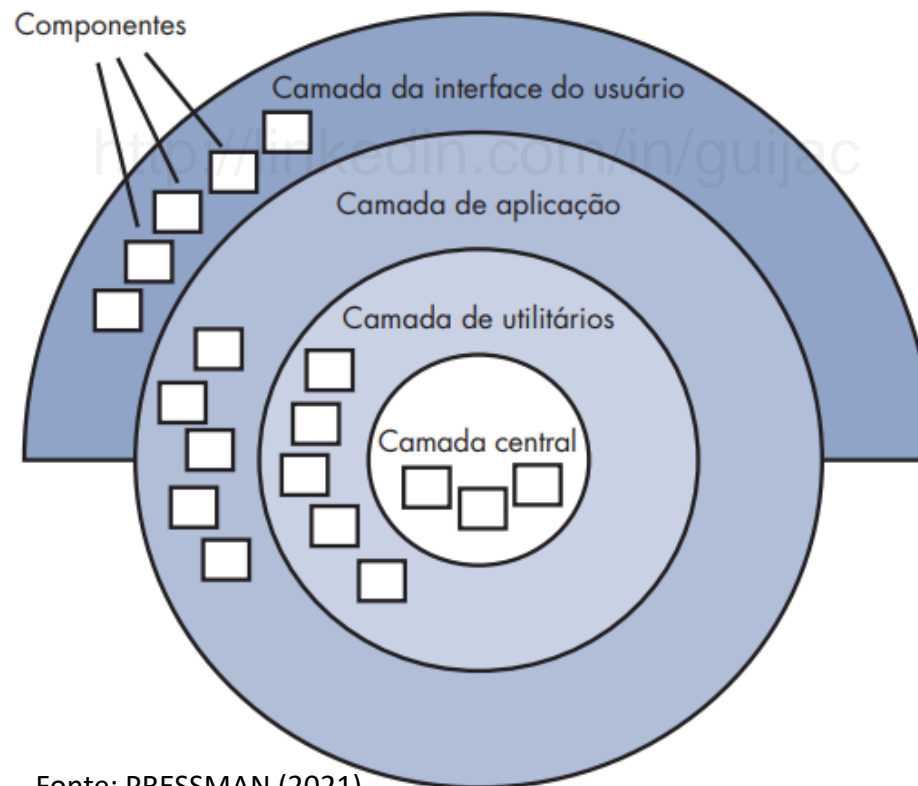
# Estilos e Padrões Arquiteturais e Padrões de Projeto (*Design Patterns*)



Fonte: Adaptado de [What is the difference between Design Pattern Vs Architecture Pattern Vs Architecture Style? - Prateek's Blog \(rprateek.com\)](https://prateek.com/what-is-the-difference-between-design-pattern-vs-architecture-pattern-vs-architecture-style/)

# Arquitetura em Camadas

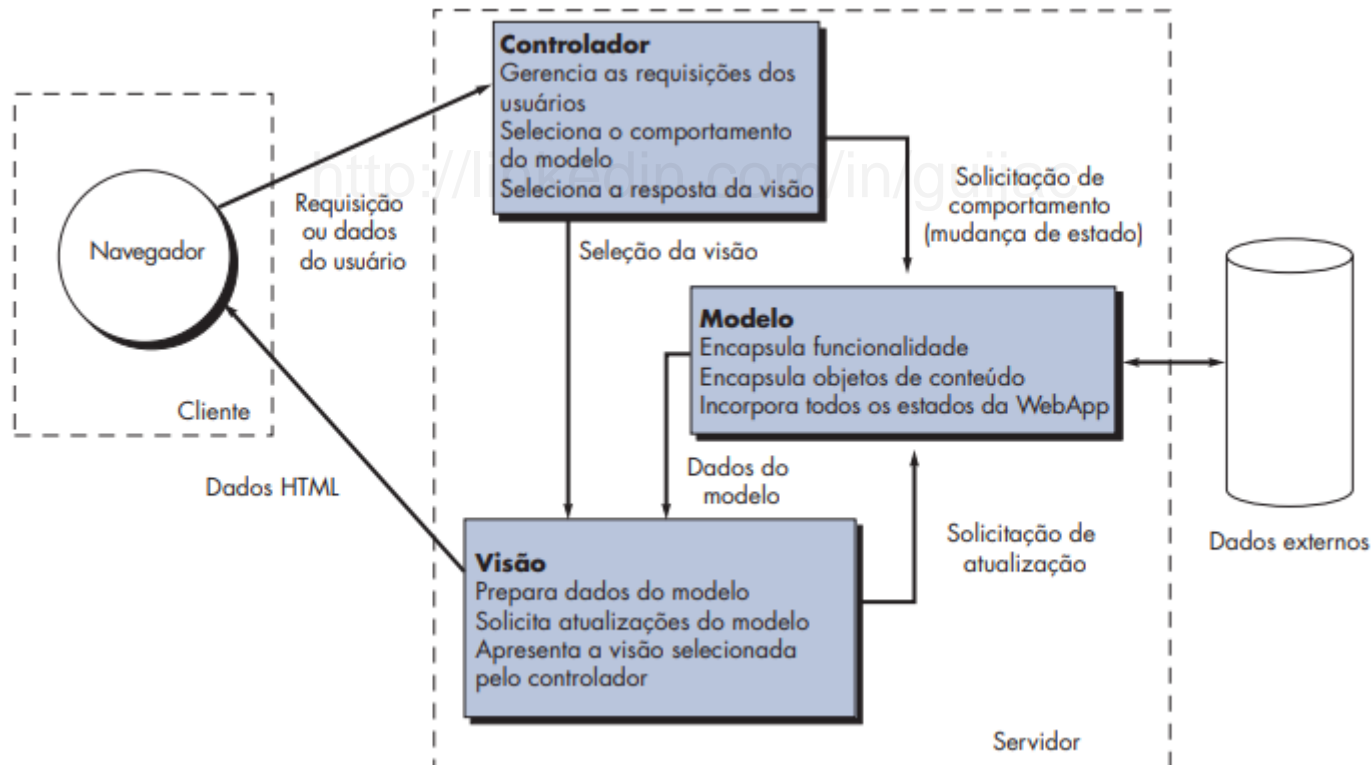
- Consiste em várias camadas, cada uma com funções progressivamente mais próximas do sistema operacional. A camada externa lida com a interface do usuário, a interna com o sistema operacional, e as intermediárias fornecem serviços e funções de aplicação. A escolha de um estilo de arquitetura irá variar conforme características e restrições do produto, podendo incluir combinações de padrões.



Fonte: PRESSMAN (2021)

# Model-View-Controller (MVC)

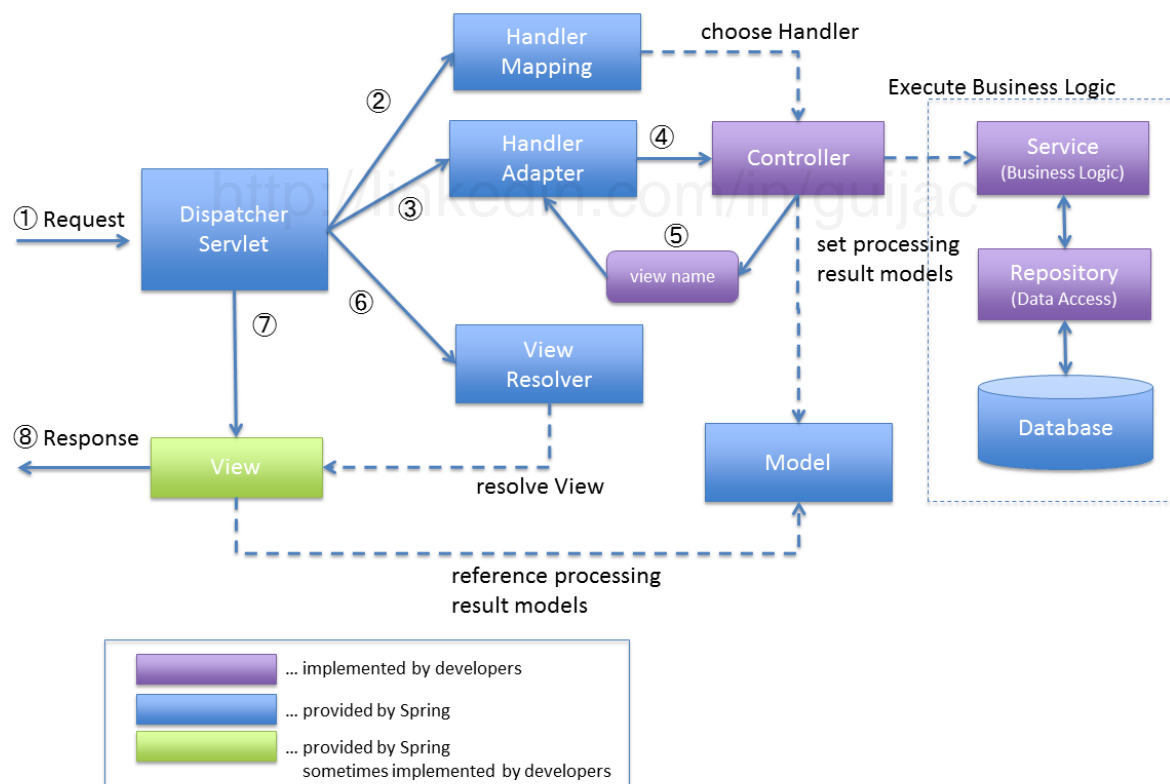
- O Modelo-Visão-Controlador (MVC) é um modelo de infraestrutura para WebApps que separa a interface do usuário da funcionalidade e do conteúdo. O **Modelo** contém a lógica da aplicação e objetos de conteúdo, a **Visão** cuida da interface e o **Controlador** coordena a comunicação entre eles.



Fonte: PRESSMAN (2021)

# Spring MVC

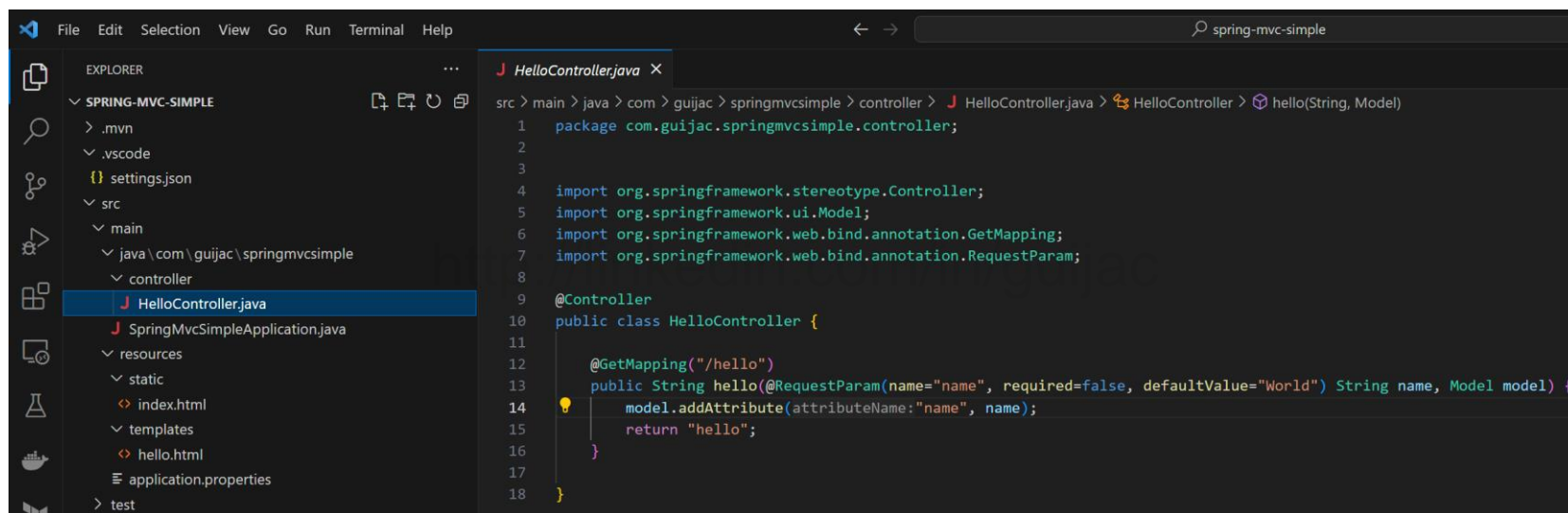
- Similar a outras estruturas MVC, projetada em torno de um Servlet que envia solicitações e oferece outras funcionalidades que facilitam o desenvolvimento de aplicações web. O DispatcherServlet do Spring, é completamente integrado ao contêiner Spring, facilitando o uso das funcionalidades do framework.



Fonte: [2.2. Overview of Spring MVC Architecture](#) — TERAOLUNA Server Framework for Java



# Spring MVC

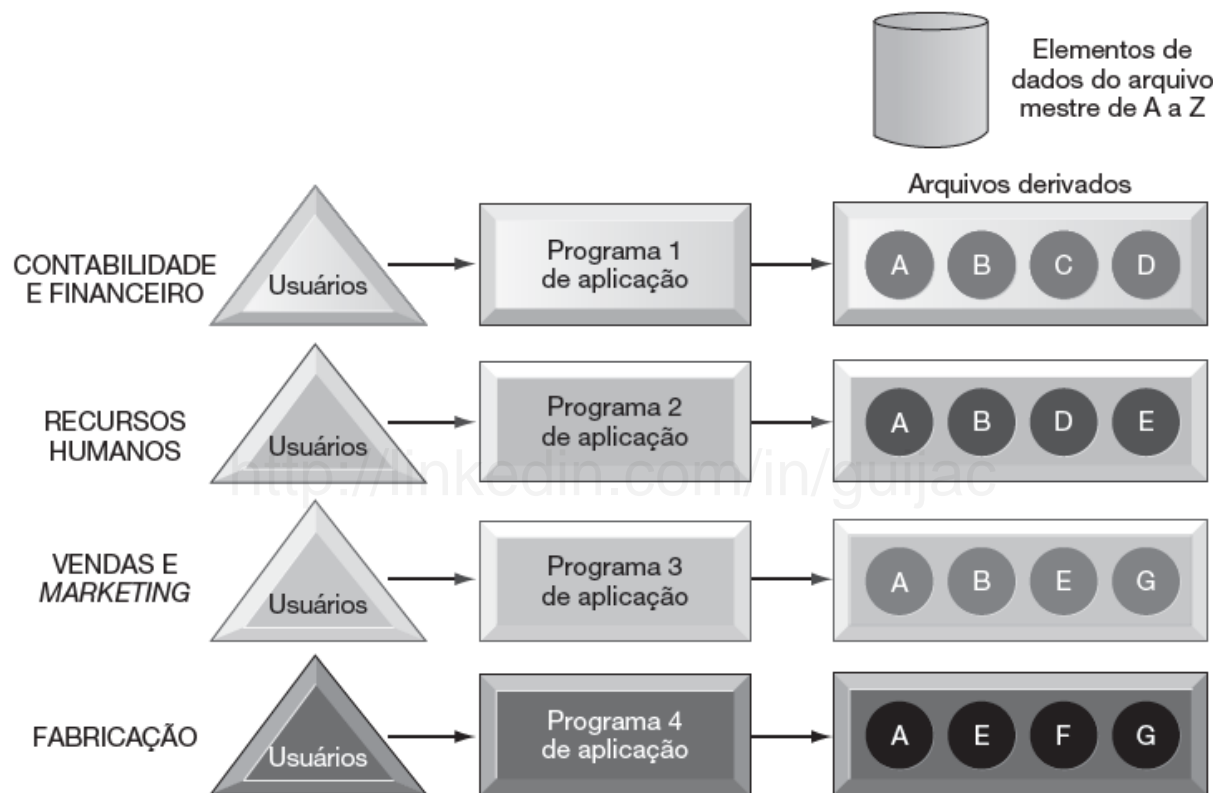


```
src > main > java > com > guijac > springmvcsimple > controller > J HelloController.java > HelloController > hello(String, Model)
1  package com.guijac.springmvcsimple.controller;
2
3
4  import org.springframework.stereotype.Controller;
5  import org.springframework.ui.Model;
6  import org.springframework.web.bind.annotation.GetMapping;
7  import org.springframework.web.bind.annotation.RequestParam;
8
9  @Controller
10 public class HelloController {
11
12     @GetMapping("/hello")
13     public String hello(@RequestParam(name="name", required=false, defaultValue="World") String name, Model model) {
14         model.addAttribute(attributeName="name", name);
15         return "hello";
16     }
17
18 }
```

Destaque de uma classe “Controller” de uma aplicação Spring MVC, gerenciando o comportamento do “Model”, com seu atributo “name” e selecionando uma resposta para a “View”, com seu conteúdo renderizado no “hello.html”.

Fonte: [guijac/spring-mvc-simple \(github.com\)](https://github.com/guijac/spring-mvc-simple)

# Arquiteturas Centralizadas em Dados



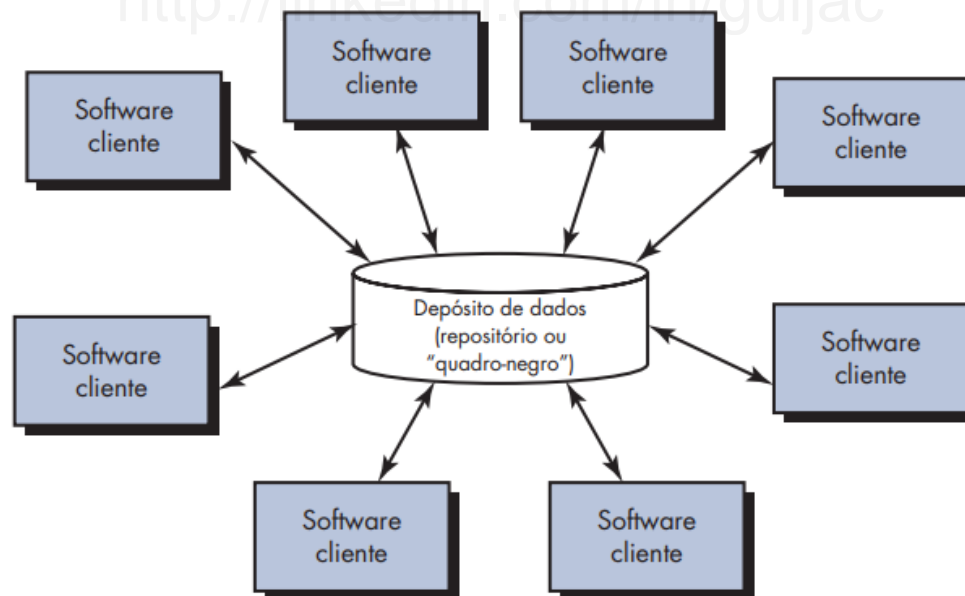
**FIGURA 6.2** Processamento de arquivos tradicional.

O uso de uma abordagem tradicional de processamento de arquivos estimula cada área funcional de uma corporação a desenvolver aplicações especializadas. Cada aplicação requer um arquivo de dados exclusivo que provavelmente será um subconjunto do arquivo mestre, levando à redundância e inconsistência de dados, inflexibilidade de processamento e desperdício de recursos de armazenamento.

Fonte: LAUDON (2019)

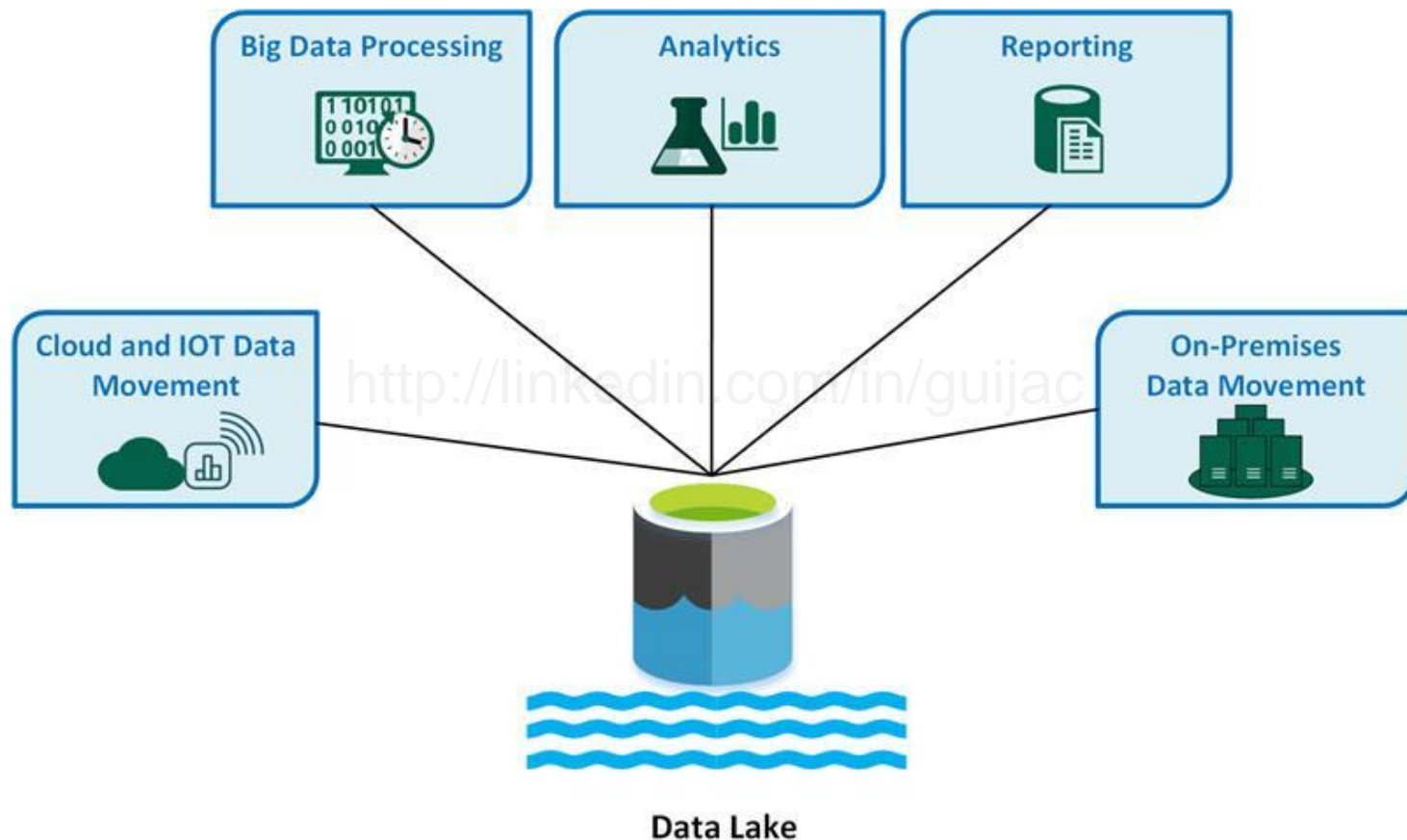
# Arquiteturas Centralizadas em Dados

- Um repositório de dados reside no centro desta arquitetura, sendo acessado frequentemente por outros componentes que atualizam, acrescentam, excluem ou modificam de alguma maneira estes dados;
- Uma variação desta abordagem permite que o repositório envie alterações aos clientes quando os dados de seu interesse mudam.



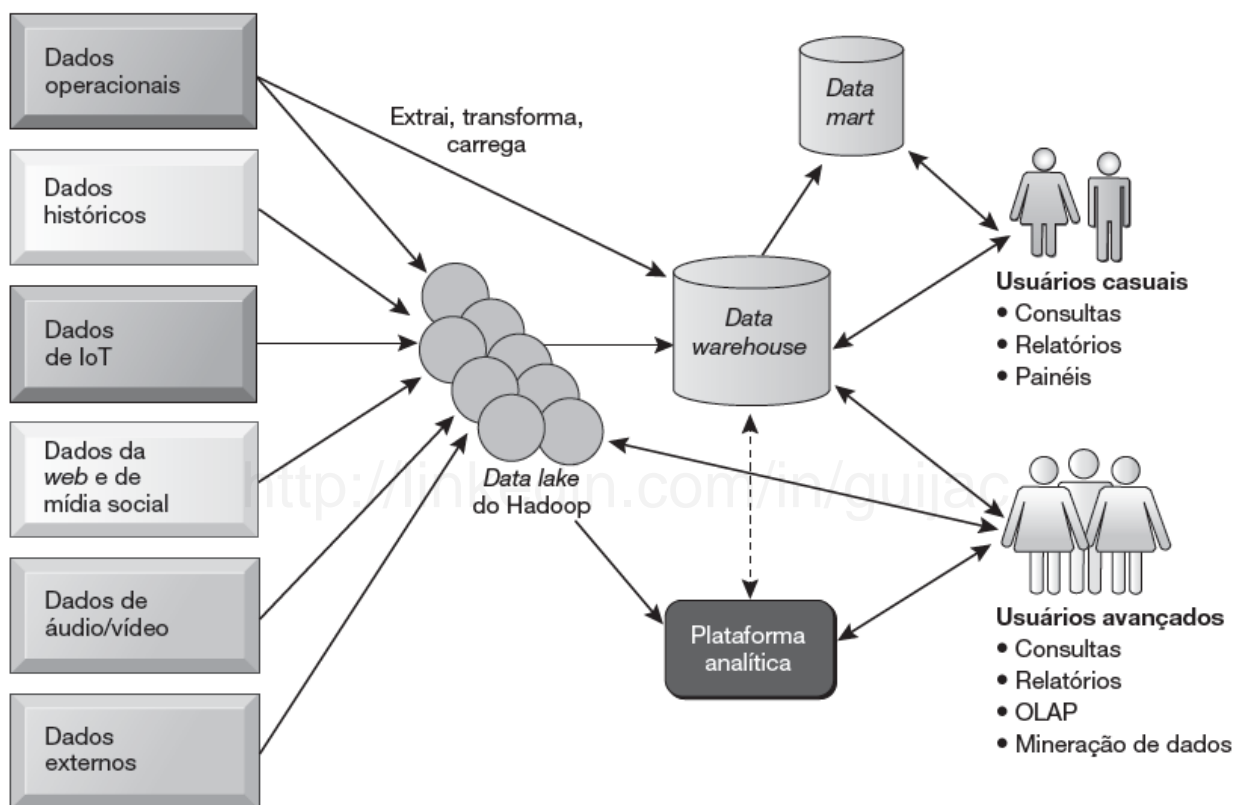
Fonte: PRESSMAN (2021)

# Arquiteturas Centralizadas em Dados



Fonte: [Data lakes - Azure Architecture Center](#) | [Microsoft Learn](#)

# Arquiteturas Centralizadas em Dados



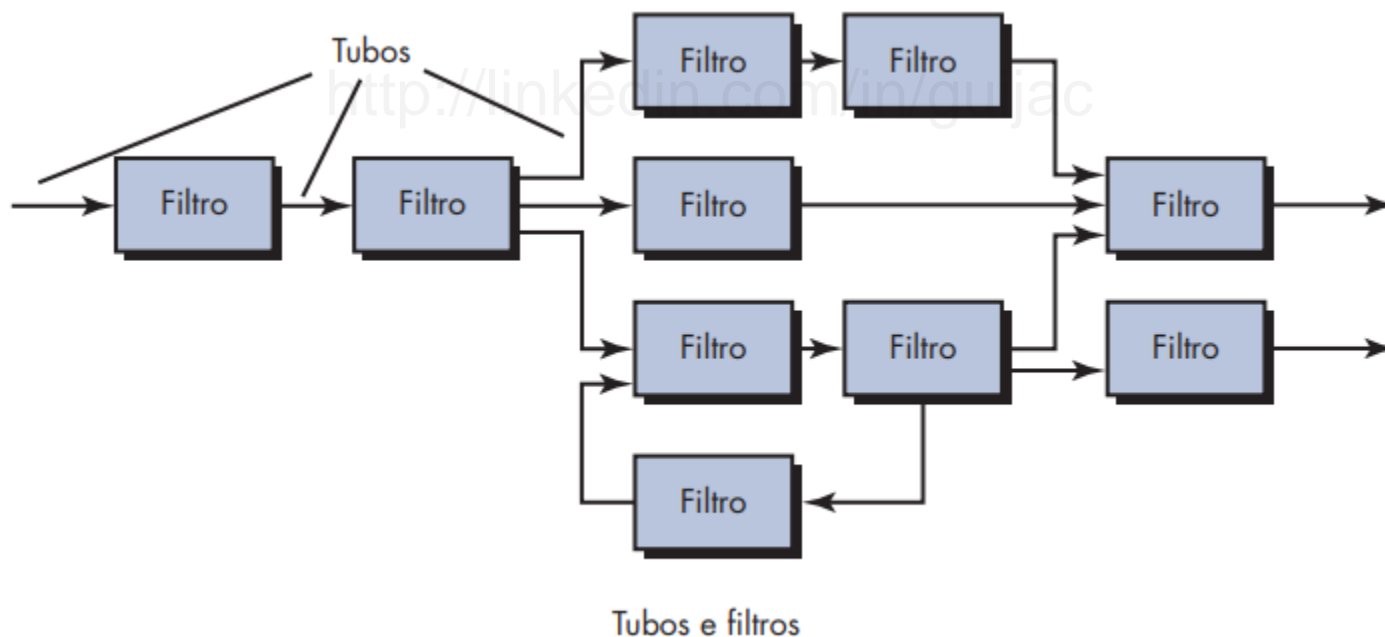
**FIGURA 6.13** Infraestrutura de *business intelligence* contemporânea.

Uma infraestrutura de *business intelligence* contemporânea apresenta recursos e ferramentas para gerenciar e analisar grandes quantidades e diferentes tipos de dados oriundos de várias fontes. Estão incluídas ferramentas de consulta e de geração de relatórios fáceis de usar para usuários de negócios casuais e conjuntos de ferramentas analíticas mais sofisticadas para usuários avançados.

Fonte: LAUDON (2019)

# Arquiteturas de Fluxo de Dados

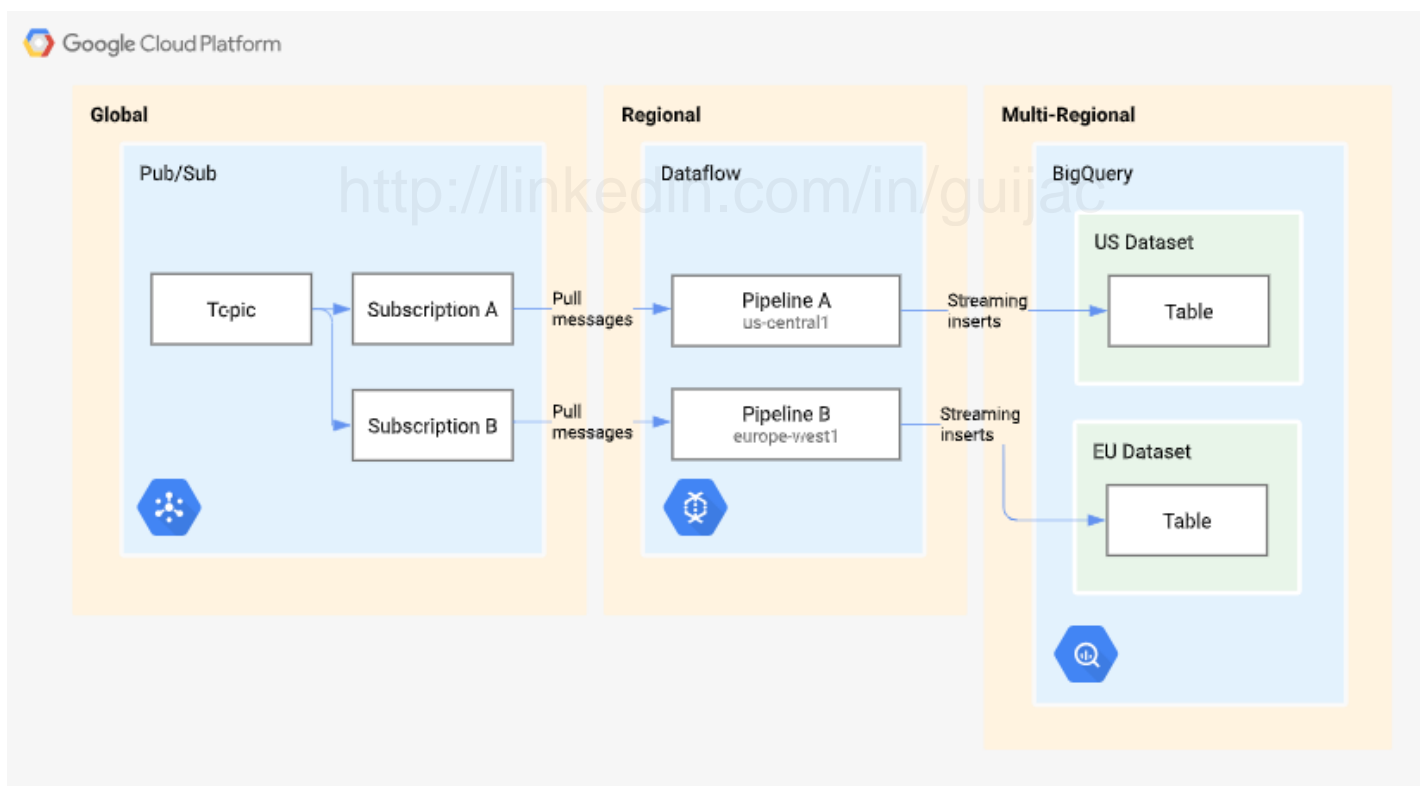
- Aplicável quando dados de entrada devem ser **transformados** por uma série de componentes computacionais;
- Utiliza-se do padrão *pipe-and-filters* (“tubos e filtros”).



Fonte: PRESSMAN (2021)

# Arquiteturas de Fluxo de Dados

- Aplicável quando dados de entrada devem ser **transformados** por uma série de componentes computacionais;
- Utiliza-se do padrão *pipe-and-filters* (“tubos e filtros”).



Fonte: [Como criar pipelines de dados prontos para produção usando o Dataflow: como implantar pipelines](#)

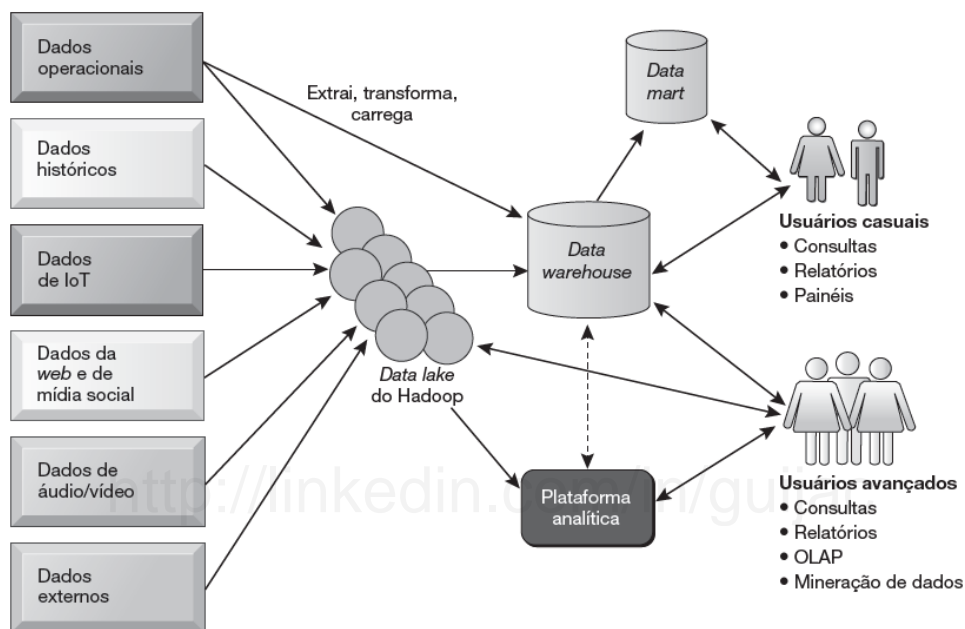
# Referências Bibliográficas

---

- ALURA. **SQL e NoSQL: trabalhando com bancos relacionais e não relacionais**. Disponível em <https://www.alura.com.br/artigos/sql-nosql-bancos-relacionais-nao-relacionais>. Acesso em 14 set 2023;
- AWS. **Teorema CAP**. Disponível em [https://docs.aws.amazon.com/pt\\_br/whitepapers/latest/availability-and-beyond-improving-resilience/cap-theorem.html](https://docs.aws.amazon.com/pt_br/whitepapers/latest/availability-and-beyond-improving-resilience/cap-theorem.html). Acesso em 14 set 2023;
- LAUDON, K. C.; LAUDON, J. P. **Sistemas De Informações Gerenciais**. 17. ed. Porto Alegre: Bookman, 2019.
- PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software-9**. McGraw Hill Brasil, 2021.



# Por hoje (de teoria) é só!



**FIGURA 6.13** Infraestrutura de *business intelligence* contemporânea.

Uma infraestrutura de *business intelligence* contemporânea apresenta recursos e ferramentas para gerenciar e analisar grandes quantidades e diferentes tipos de dados oriundos de várias fontes. Estão incluídas ferramentas de consulta e de geração de relatórios fáceis de usar para usuários de negócios casuais e conjuntos de ferramentas analíticas mais sofisticadas para usuários avançados.

Fonte: LAUDON (2019)

**Prof. Esp. Guilherme Jorge Aragão da Cruz**

 [guilherme.jacruz@sp.senac.br](mailto:guilherme.jacruz@sp.senac.br)

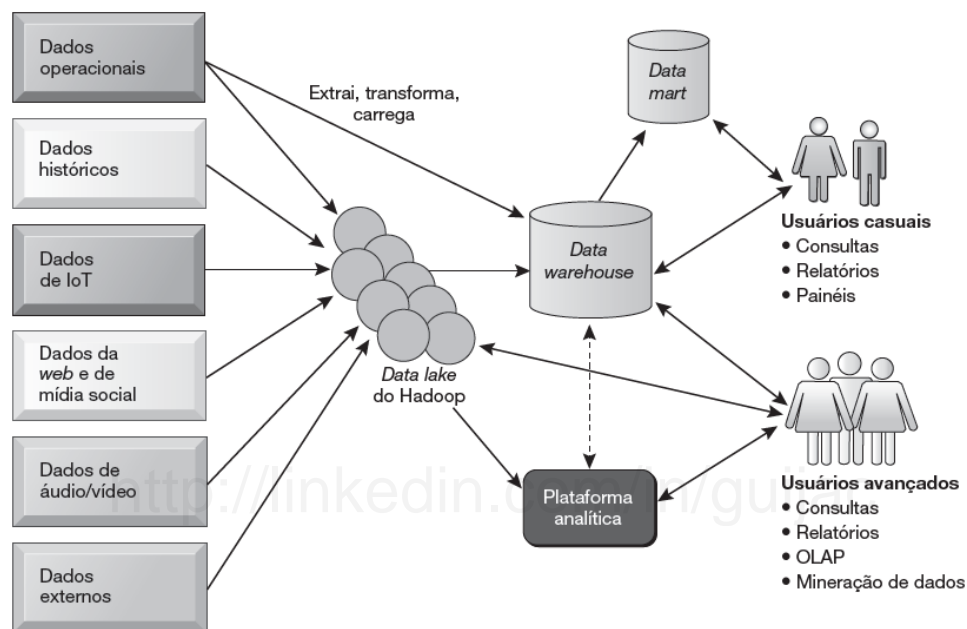
 [linkedin.com/in/guijac](https://linkedin.com/in/guijac)

# Laboratório

---

- **Em grupos**, a partir do Caso de Negócio da ArchiSurance ou através da base selecionada na ADO 01, elaborar a arquitetura que originou a base analítica (.csv):
  - Propor Aplicações (quantas aplicações originaram cada conjunto de dado?);
  - Propor Modelo(s) e plataforma para Banco de Dados (relacional/não relacional Oracle, SQL Server, MongoDB, Cassandra, etc);
  - Propor Padrão/Padrões Arquiteturais para Aplicações (centralizada em dados, de fluxo de dados).
- Sugestão de ferramenta para desenho: [Flowchart Maker & Online Diagram Software](#)

# Por hoje (agora sim) é só!



**FIGURA 6.13** Infraestrutura de *business intelligence* contemporânea.

Uma infraestrutura de *business intelligence* contemporânea apresenta recursos e ferramentas para gerenciar e analisar grandes quantidades e diferentes tipos de dados oriundos de várias fontes. Estão incluídas ferramentas de consulta e de geração de relatórios fáceis de usar para usuários de negócios casuais e conjuntos de ferramentas analíticas mais sofisticadas para usuários avançados.

Fonte: LAUDON (2019)

**Prof. Esp. Guilherme Jorge Aragão da Cruz**

 guilherme.jacruz@sp.senac.br

 linkedin.com/in/guijac