



UNIVERSIDADE  
LUSÓFONA

# Base de Dados

Apresentação, introdução e ferramentas

2022/2023

# Sumário

Apresentação

Introdução às Bases de Dados e ao SQL

Instalação do ambiente de trabalho

- Servidor
- Cliente
- Docker



Base de Dados?

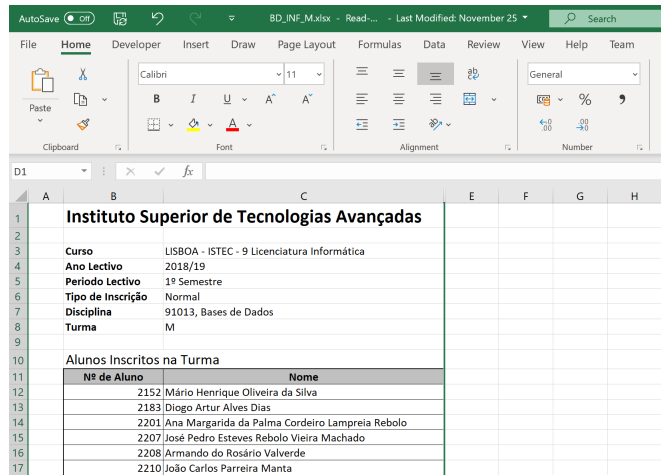


# Base de Dados?

Uma base de dados é:

- um repositório de informação
- sobre determinado assunto ou finalidade
- uma colecção de dados ou itens informação estruturados
- permite a sua consulta, actualização e processamento através de meios informáticos.

# Alguns Exemplos de Bases de Dados



The screenshot shows an Excel spreadsheet with a database for 'Instituto Superior de Tecnologias Avançadas'. The data is organized into two main sections: 'Curso' (Course) and 'Alunos Inscritos na Turma' (Students Registered in the Class).

Curso	
Curso	LISBOA - ISTE - 9 Licenciatura Informática
Ano Lectivo	2018/19
Período Lectivo	1º Semestre
Tipo de Inscrição	Normal
Disciplina	91013, Bases de Dados
Turma	M

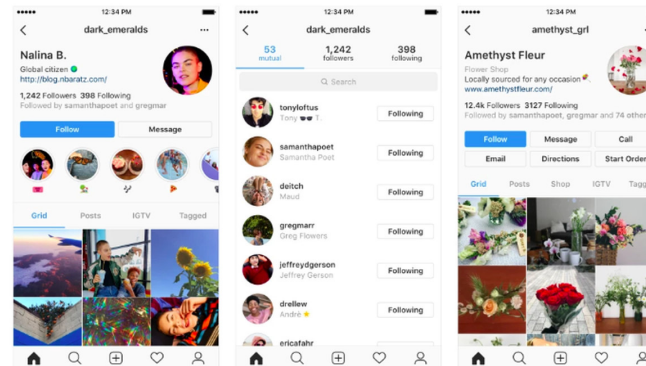
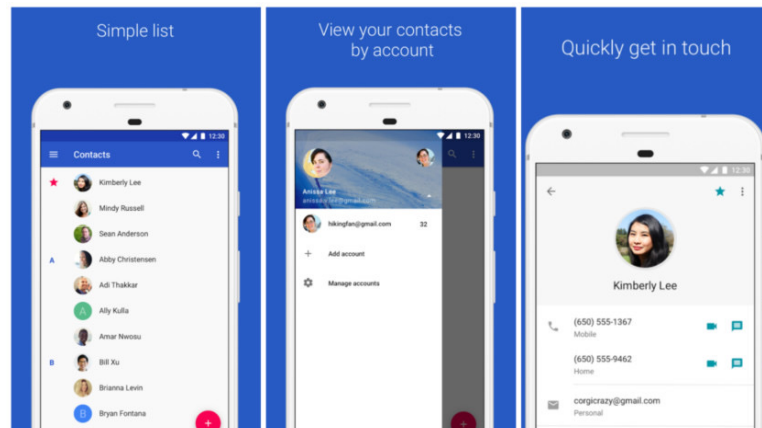
  

Alunos Inscritos na Turma	
Nº de Aluno	Nome
2152	Mário Henrique Oliveira da Silva
2183	Diogo Artur Alves Dias
2201	Ana Margarida da Palma Cordeiro Lampreia Rebole
2207	José Pedro Esteves Rebole Vieira Machado
2208	Armando do Rosário Valverde
2210	João Carlos Parreira Manta

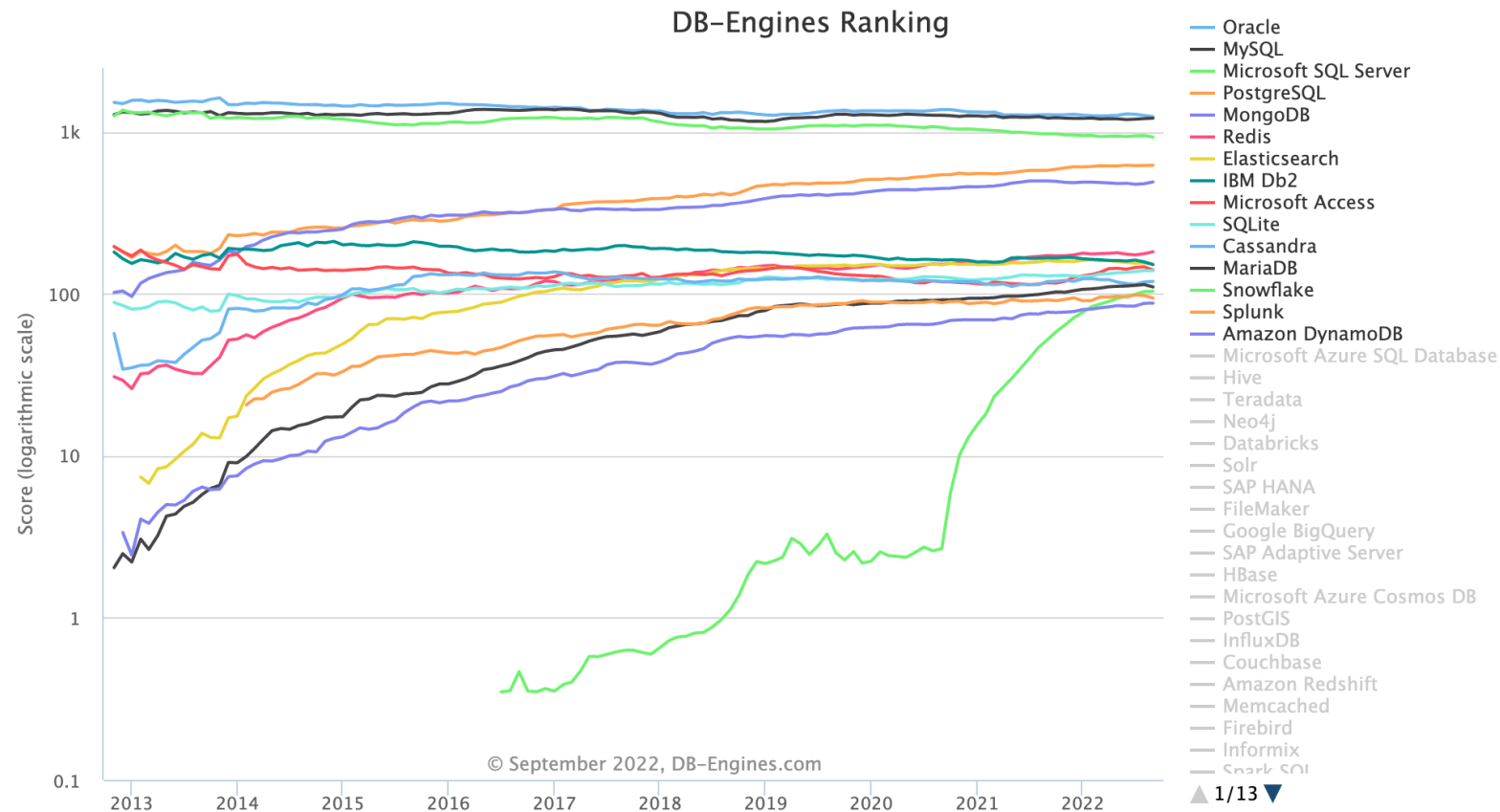


The screenshot shows the Caixa Geral de Depósitos website. It displays the account type 'Checking Business' and 'Checking Personal'. Below each account type, there are links to view activity, statements, pay bills, and routing/account numbers. The 'Savings Taxes' section also shows links for activity, statements, and routing/account numbers. The 'Savings Emergency' and 'Savings Investment' sections show links for activity, statements, and routing/account numbers. The 'Savings Tithes' and 'Savings Personal' sections show links for activity, statements, and routing/account numbers.

Account	Balance	Present Balance
Checking Business	12,111.11	12,111.11
Checking Personal	12,111.11	12,111.11
Savings Taxes	12,111.11	12,111.11
Savings Emergency	12,111.11	12,111.11
Savings Investment	12,111.11	12,111.11
Savings Tithes	12,111.11	12,111.11
Savings Personal	12,111.11	12,111.11



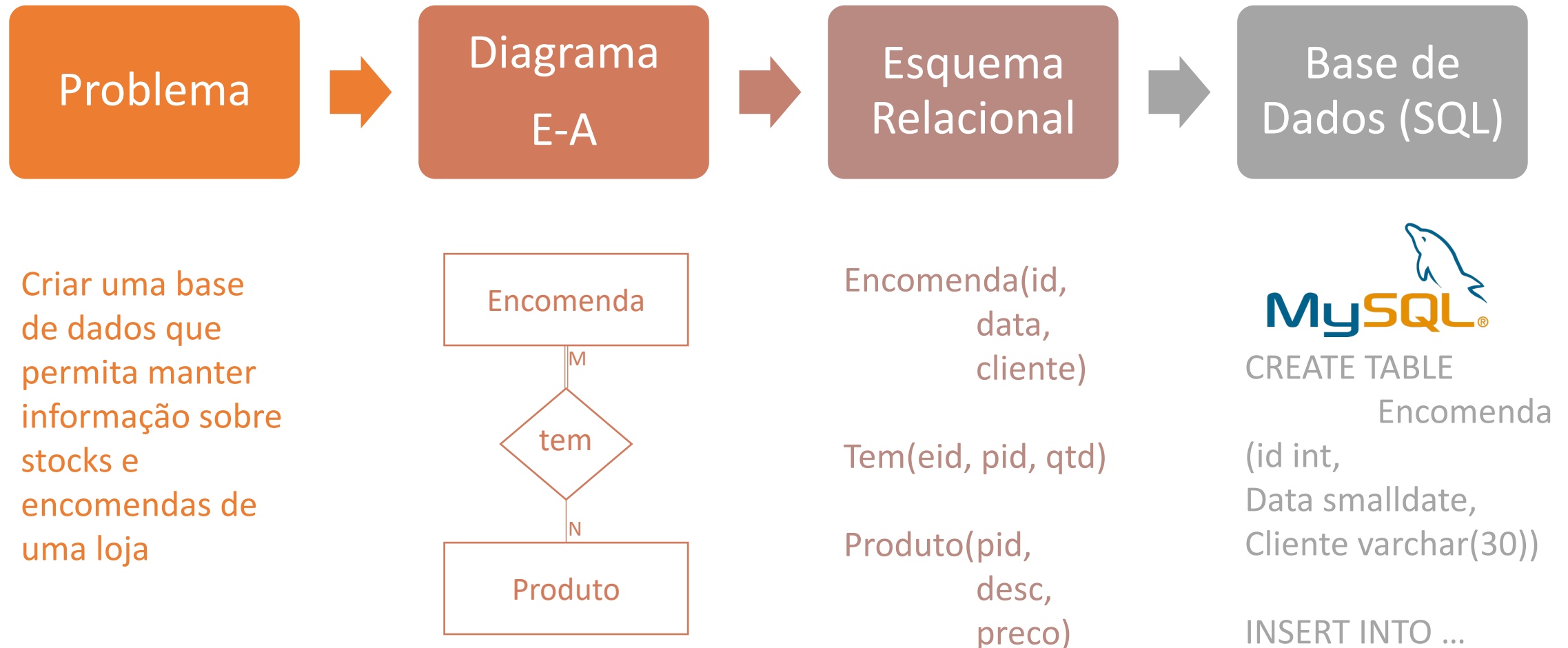
# Ranking de Sistemas de Base de Dados





Apresentação

# Processo de Criação de uma BD Relacional





# Funcionamento das aulas



Correção trabalho de casa / Dúvidas  
sobre aula anterior



Apresentação da matéria



Exercícios



Trabalho de casa

# Avaliação

## Projeto de Base de Dados (grupos de 3 alunos)

- 1ª entrega: 30%
- 2ª entrega: 35%
- 3ª entrega: 35%

## Avaliação contínua (individual +/-10%)

Nota mínima 9.5v

# Canais de Comunicação

## Moodle

- Avisos
- Disponibilização de material
- Entrega de trabalhos
- Publicação de avaliações

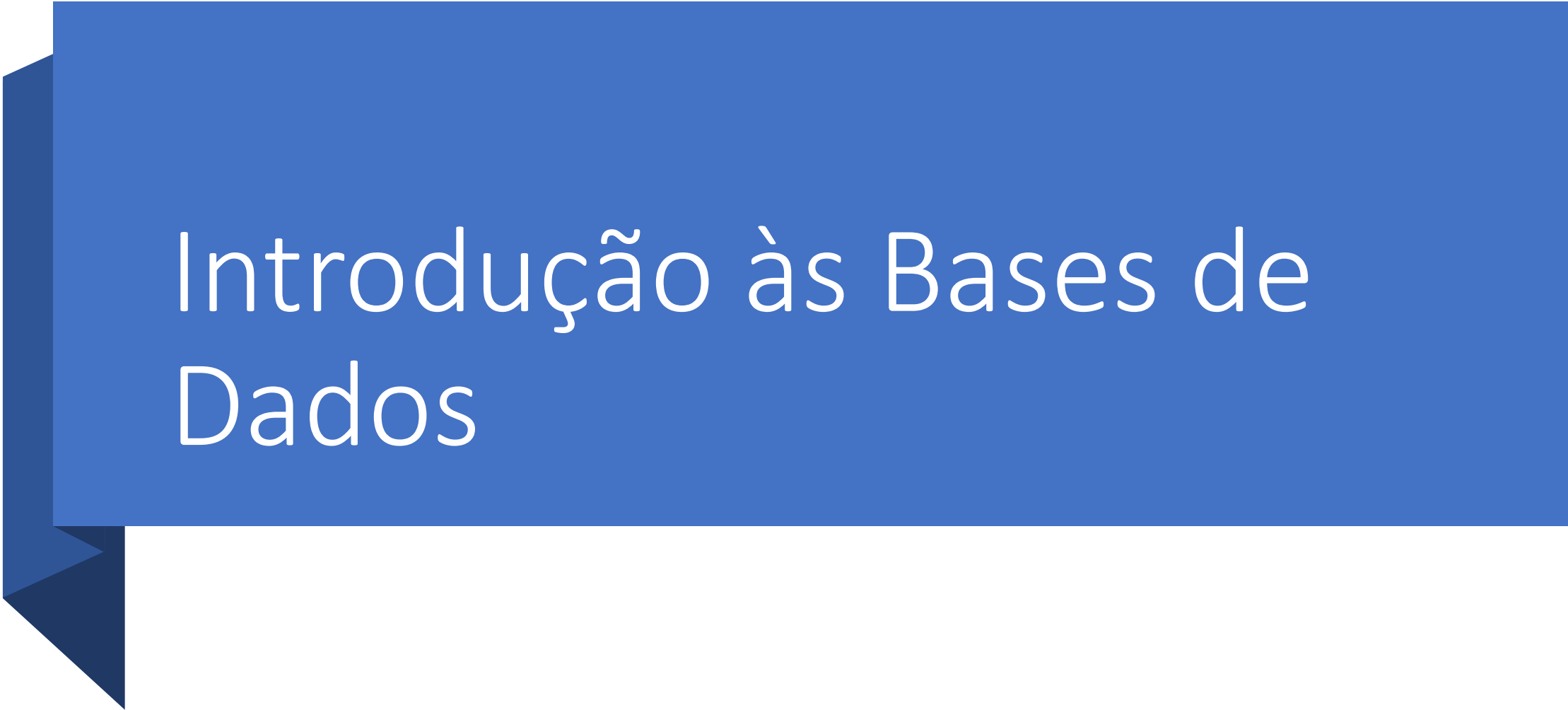
## GitHub

## Slack

- Dúvidas sobre funcionamento da disciplina
- Dúvidas sobre matéria, trabalhos
- Discussão de tópicos diversos

Email ([jcarneiro.ulht@gmail.com](mailto:jcarneiro.ulht@gmail.com))

- Outros assuntos



# Introdução às Bases de Dados

# Sumário

Bases de Dados

Sistemas de Gestão de  
Base de Dados

Conceitos-chave

# Base de Dados em Ficheiro

1960

1970

1980

1990

2000

2010

2020



- Armazenar a nossa base de dados em ficheiros de valores separados por vírgula (CSV – comma-separated value) geridos no nosso próprio código.
- Criar um ficheiro para cada entidade
- A aplicação tem de processar os ficheiros cada vez que se quer ler ou atualizar os dados

# Base de Dados em Ficheiro

- Criar uma base de dados de contactos (nome, apelido, telefone)

**Contactos.txt** (nome, apelido, telefone)

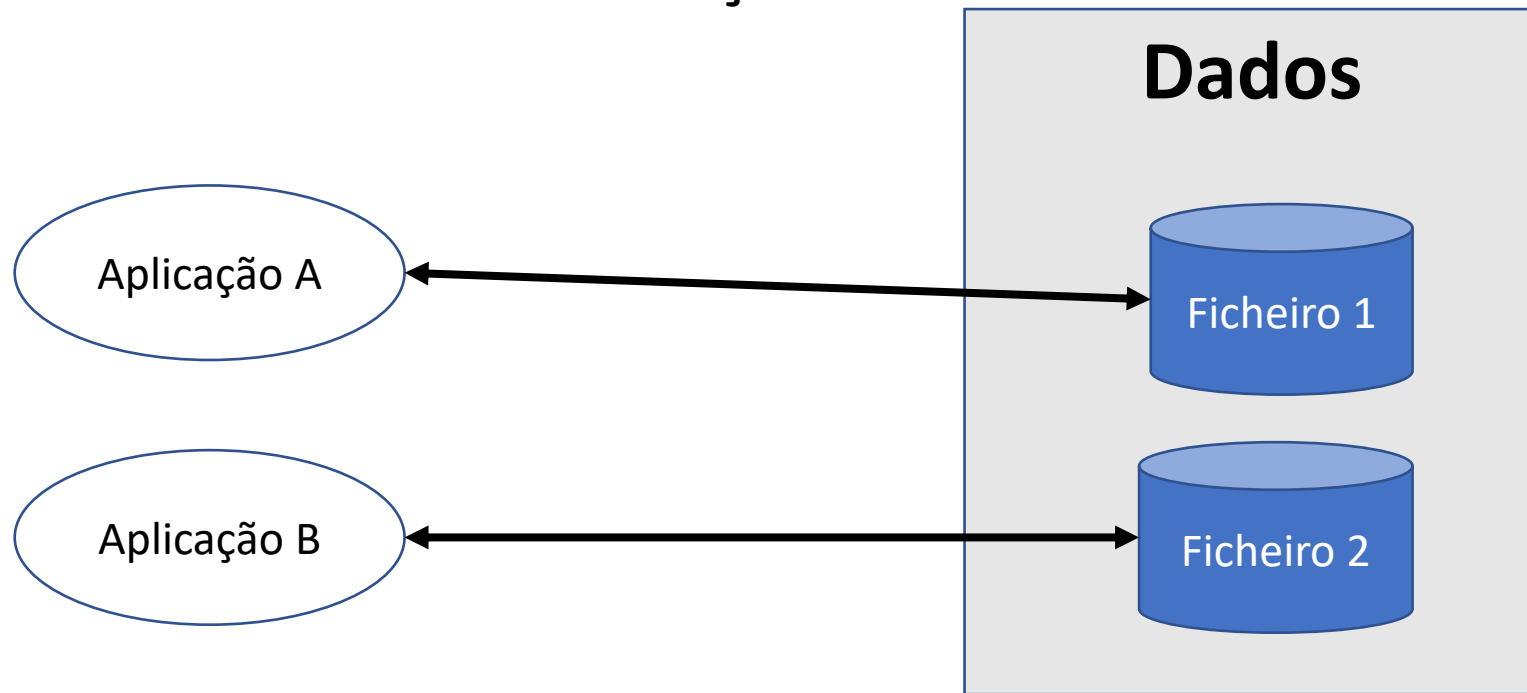
Luisa, Martins, 918654823

Joana, Reis, 969178845

Pedro, Henriques, 931564412

# Sistemas de Armazenamento de Dados

- **Sistemas de Ficheiros** - Cada aplicação mantém os ficheiros com dados necessários à sua execução





# Primeiros SGBD (anos 1960)

1960

1970


1980

1990

2000

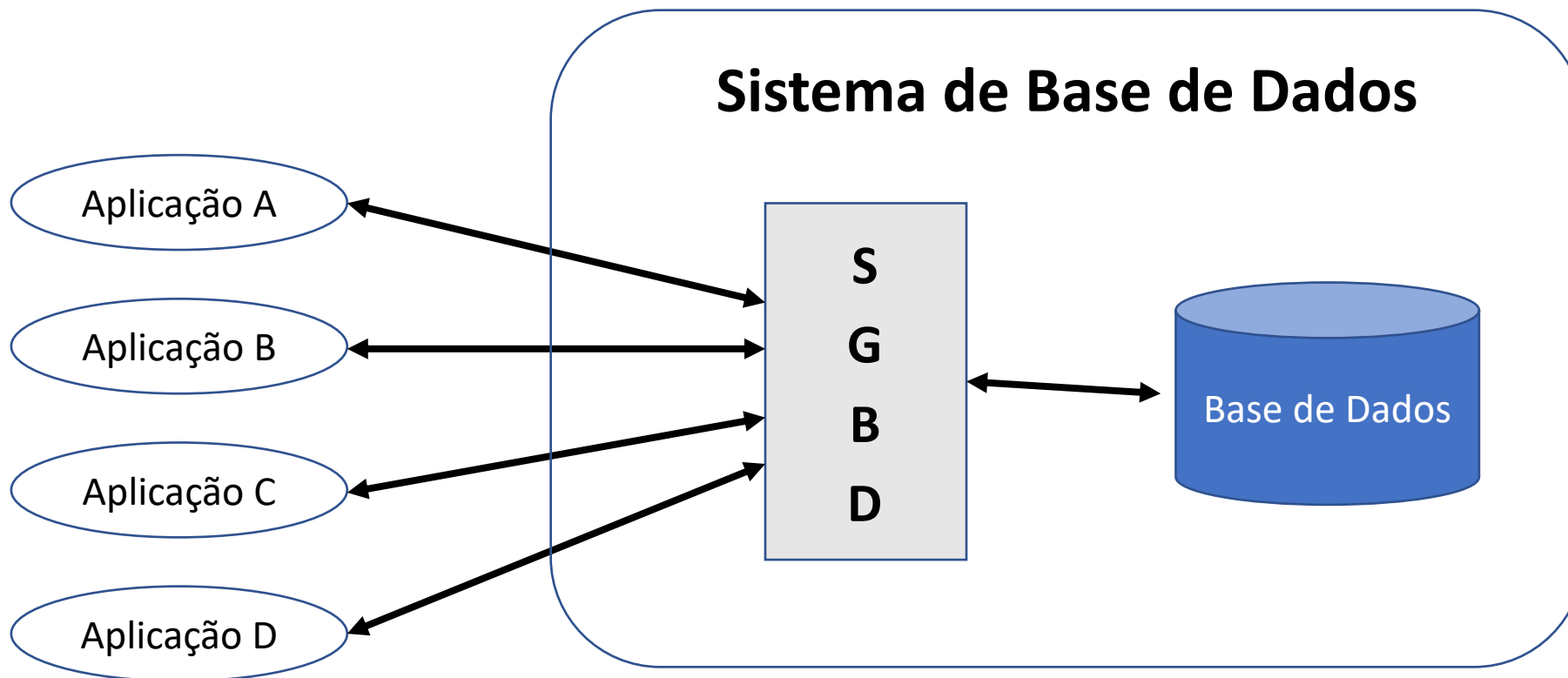
2010

2020

- 
- Aplicações com base de dados eram difíceis de construir e manter
  - Ligação intrínseca entre camadas lógica e física
  - Necessário conhecer as pesquisas (queries) que a aplicação vai fazer antes de desenvolver a base de dados

# Primeiros SGBD (anos 1960)

Oferecem uma forma eficiente, fiável, conveniente e segura de acesso e armazenamento por multi-utilizadores de grandes quantidades de dados persistentes



# História das BD (anos 1970)

1960

**1970**

1980

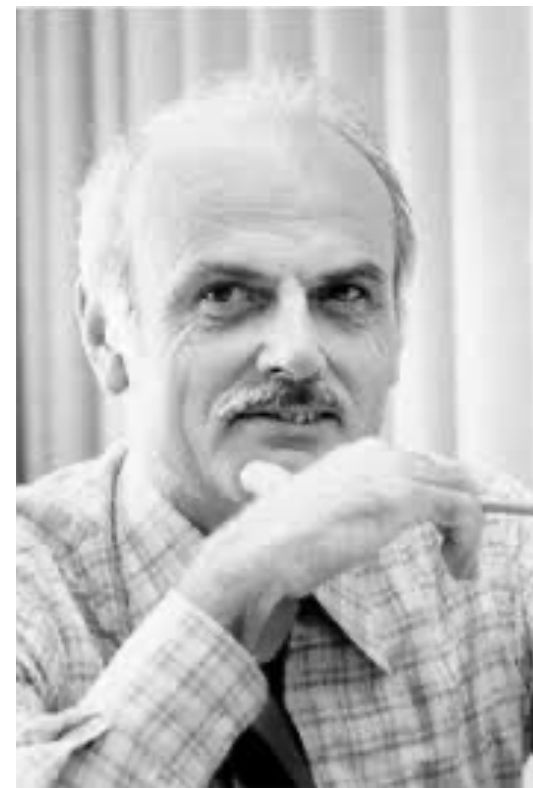
1990

2000

2010

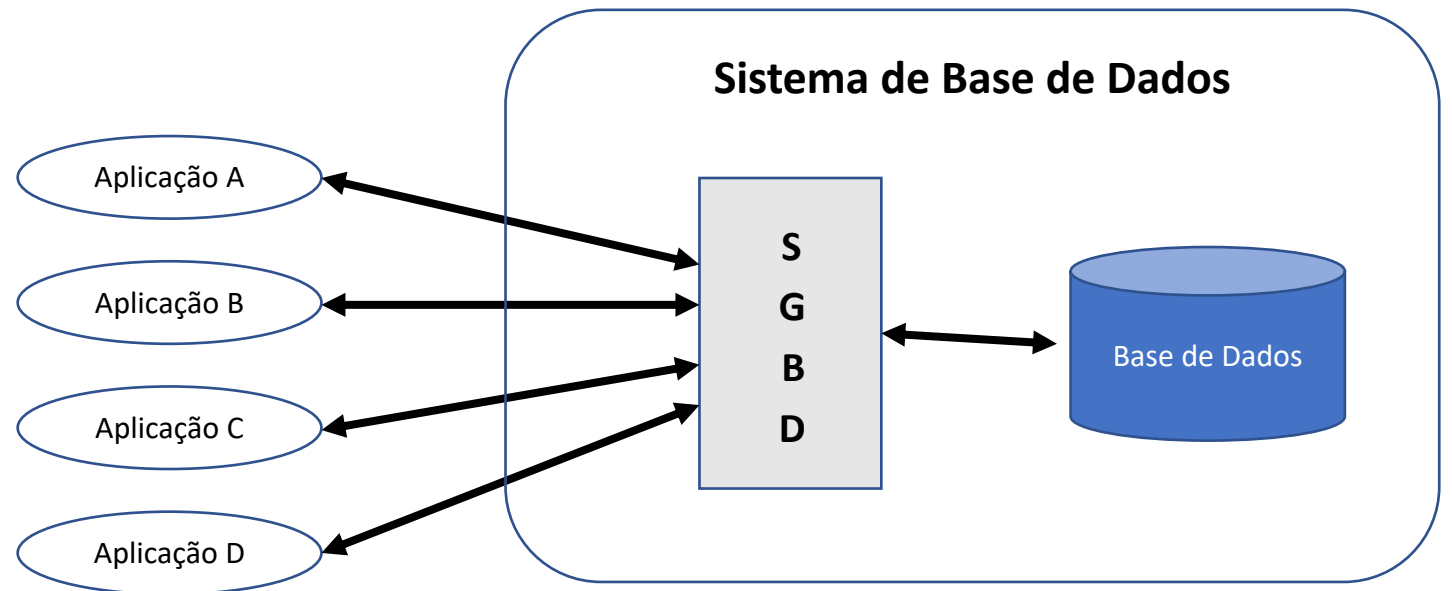
2020

- **Modelo Relacional**
- Proposto por Edgar Codd
- Abstração da base de dados para evitar manutenção
  - Armazenar base de dados em estruturas de dados simples
  - Aceder aos dados através de uma linguagem de alto nível
  - Armazenamento físico responsabilidade da implementação do SGBD



# SGBD - Características

- Grandes quantidade de Dados
- Persistente
- Segurança
- Multi-utilizador
- Conveniente
- Eficiente
- Fiável/Disponível



# Sistema de Gestão de Base de Dados

- Conjunto de programas
- Armazenamento e manipulação de dados
- Fornece dados a programadores e utilizadores
- Controlo de acesso
- Independência dos dados

# Modelo de Armazenamento de Dados

- Existem várias possibilidades de estruturar informação
- Descrição da estrutura dos dados
  - Relacional - Conjunto de registos
  - Hierarquico – XML
  - Grafos – nós e ligações

Relational World

Branch					
ID	Branch	Type	Count	SalesAssistant	Product
2313	Boston	Shoes	36	324	756
2654	Worcester	Shoes	39	354	567
6546	Weymouth	ShoeCareProducts	127	654	445

Documents - JSON

```
{
  "type": "sweater",
  "color": "blue",
  "size": "M",
  "material": ["wool", "cotton"],
  "form": "turtleneck"
}

{
  "type": "sweater",
  "color": "blue",
  "size": ["S", "M", "L"],
  "material": "wool",
  "form": "turtleneck"
}

{
  "type": "pants",
  "waist": 32,
  "length": 34,
  "color": "blue",
  "material": "cotton"
}

{
  "type": "television",
  "diagonal screen size": 46,
  "hdmi inputs": 3,
  "wall mountable": true,
  "built-in digital tuner": true,
  "dynamic contrast ratio": "50,000:1",
  "Resolution": "1920x1080"
}
```

Graphs

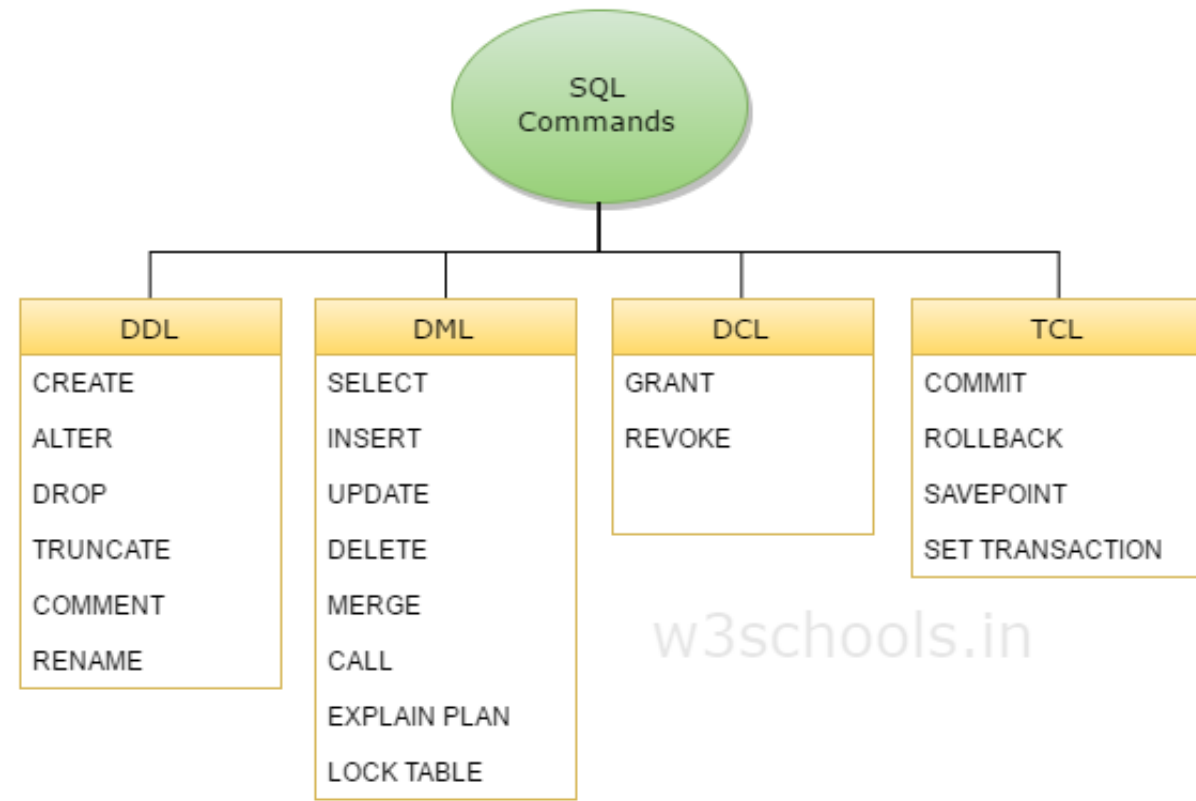


# Esquemas e Instâncias

- Equivalente a tipos e variáveis em programação
- Esquema – estrutura lógica de uma base de dados
  - Análogo ao tipo de variável
- Instância – o conteúdo da base de dados num determinado momento
  - Análogo ao valor de uma variável

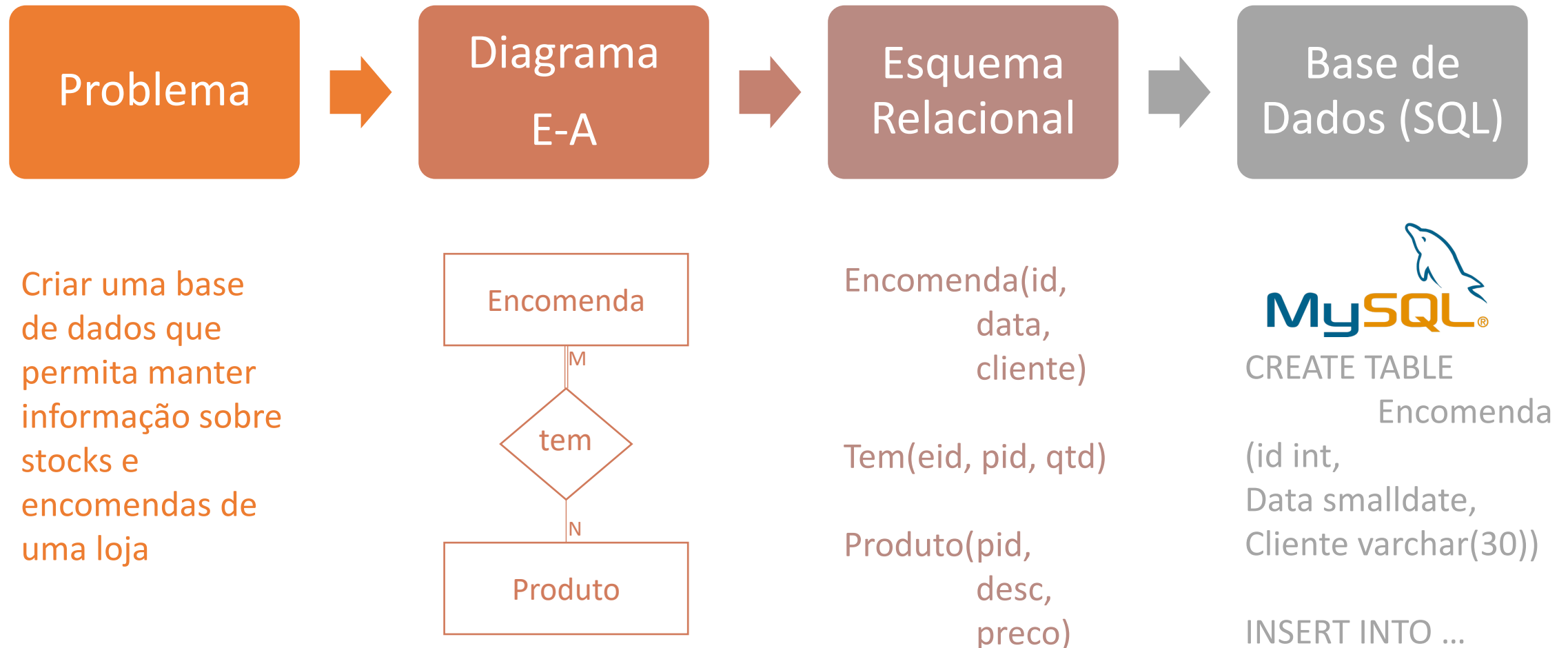
# SQL – Structured Query Language

- Linguagem declarativa
- Composta por várias linguagens
  - Linguagem de Definição de Dados (DDL)
    - Especificação do esquema da base de dados
  - Linguagem de Manipulação dos Dados (DML)
    - Linguagem para acesso e manipulação dos dados
  - Linguagem de Controlo de Dados (DCL)
    - Operações de segurança e controlo de acessos
  - Linguagem de Controlo de Transações (TCL)
    - Manipulação de transações





# Processo de Criação de uma BD Relacional



A blue ribbon graphic with a 3D effect, featuring a lighter blue top surface and a darker blue bottom surface, framing the text.

# Ferramentas Bases de Dados

# Arquitetura Cliente-Servidor

```
> mysql -h 192.168.0.215 -ujose -p
```



Server  
192.168.0.215

## Verificação ligação

username: jose

password: 1234\$

jose@192.168.0.100



PC

192.168.0.100

jose@192.168.0.101



Smartphone

192.168.0.101

jose@192.168.0.102



Laptop

192.168.0.102

# Servidores

- Alguns exemplos conhecidos

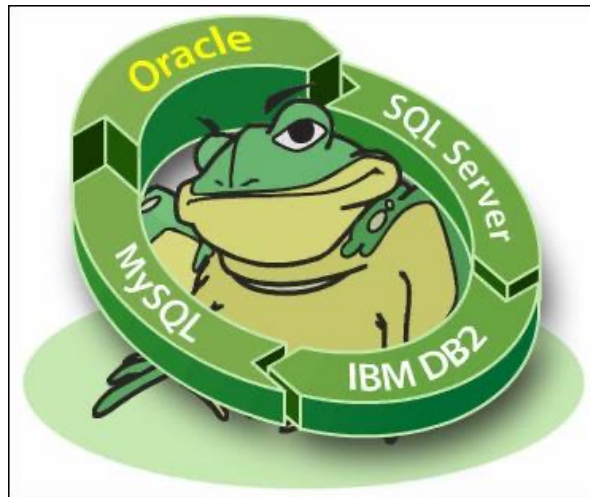


ORACLE



teradata.

# Clientes



# Introdução ao Docker

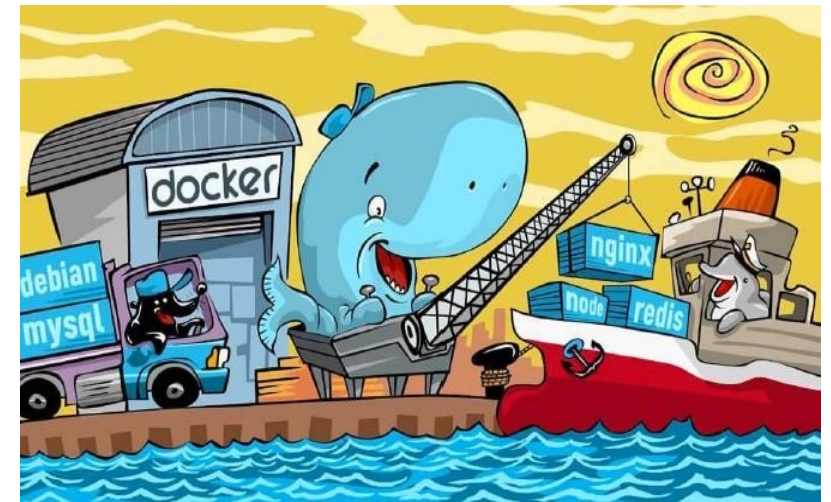
(slides Prof Luís Gomes)

# O que é Docker?

**Docker** é um programa de computador que realiza operações de virtualização de sistemas operativos, estas são mais conhecidas como "containerization";

**Docker** permite empacotar uma aplicação dentro de uma unidade *standard* para o desenvolvimento de software – *The Docker container*.

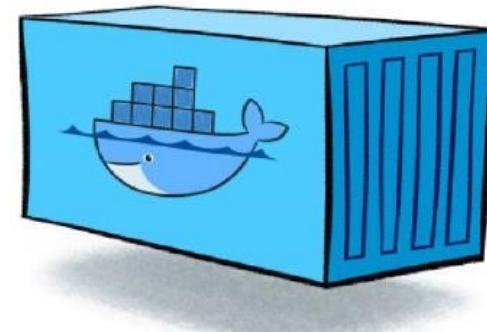
**Docker promise:** *Build, Ship and Run!*



# O que é um Container?

*Container* é um pacote de código que contem todos os elementos necessários para ser executável sem erros;

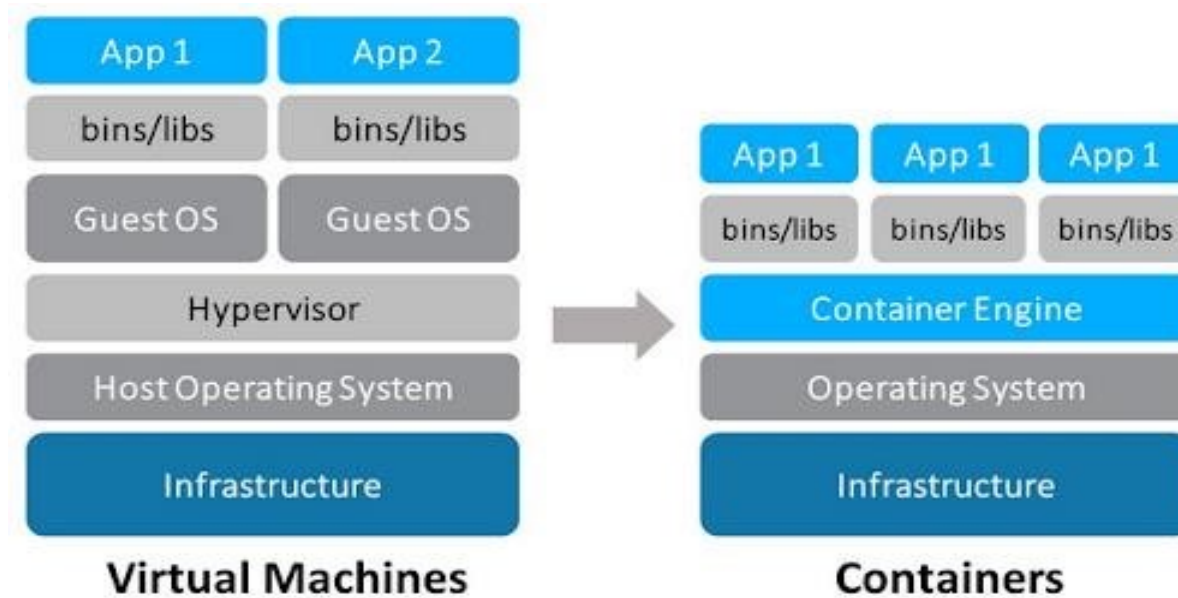
- Estes elementos são constituídos por:
- Ferramentas dos sistema;
- *Libraries*;
- Configurações.





# O que é um Container?

Um *Container* funciona como uma camada de virtualização para serviços ou aplicações



*Containerização não é virtualização*

Não há isolamento do ambiente virtual

# Containers: vantagens e desvantagens

## Vantagens

- Ambiente *Agile*;
- Aumento de produtividade;
- Controlo de versões;
- Portabilidade do ambiente computacional;
- *Standardization*;
- Segurança.

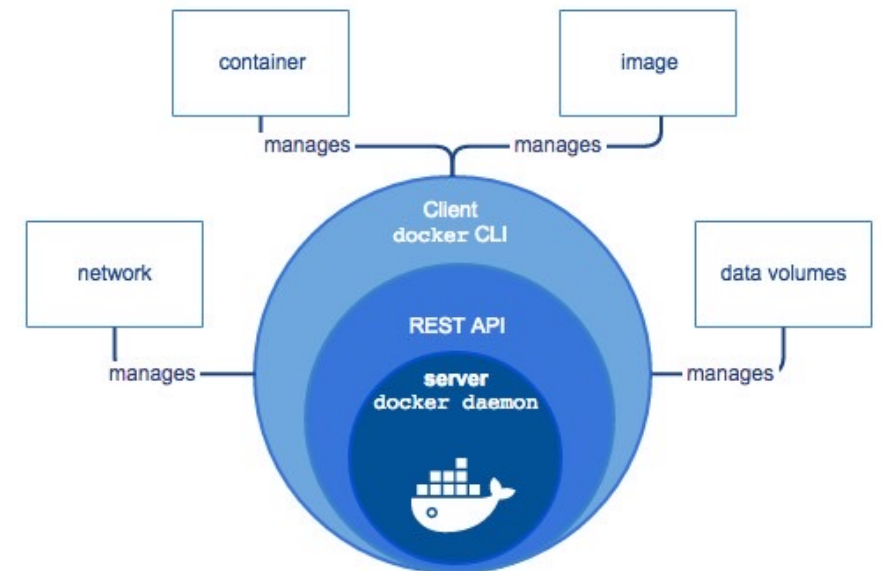
## Desvantagens

- Complexidade aumentada;
- Informação privada dentro dos *containers*;
- Data persistida permanente.

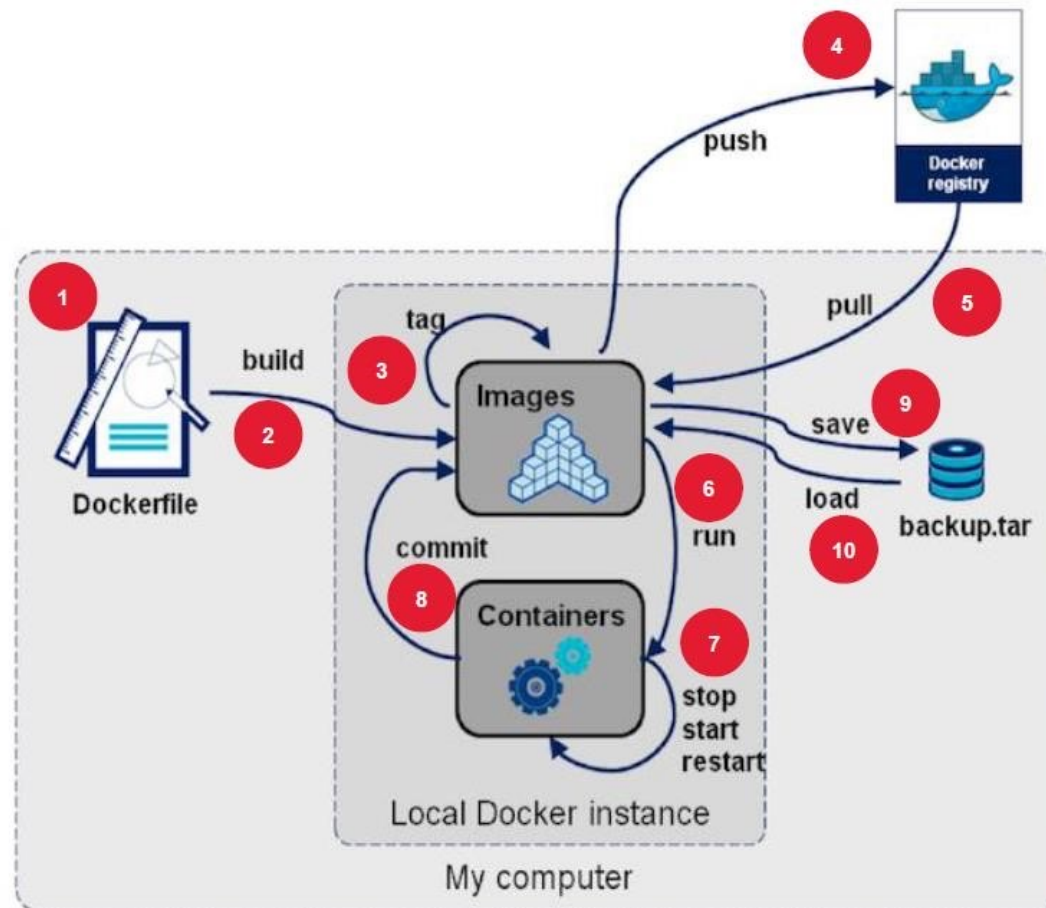
# Arquitetura

*Docker Engine* é uma aplicação cliente-servidor constituída por 3 elementos fundamentais:

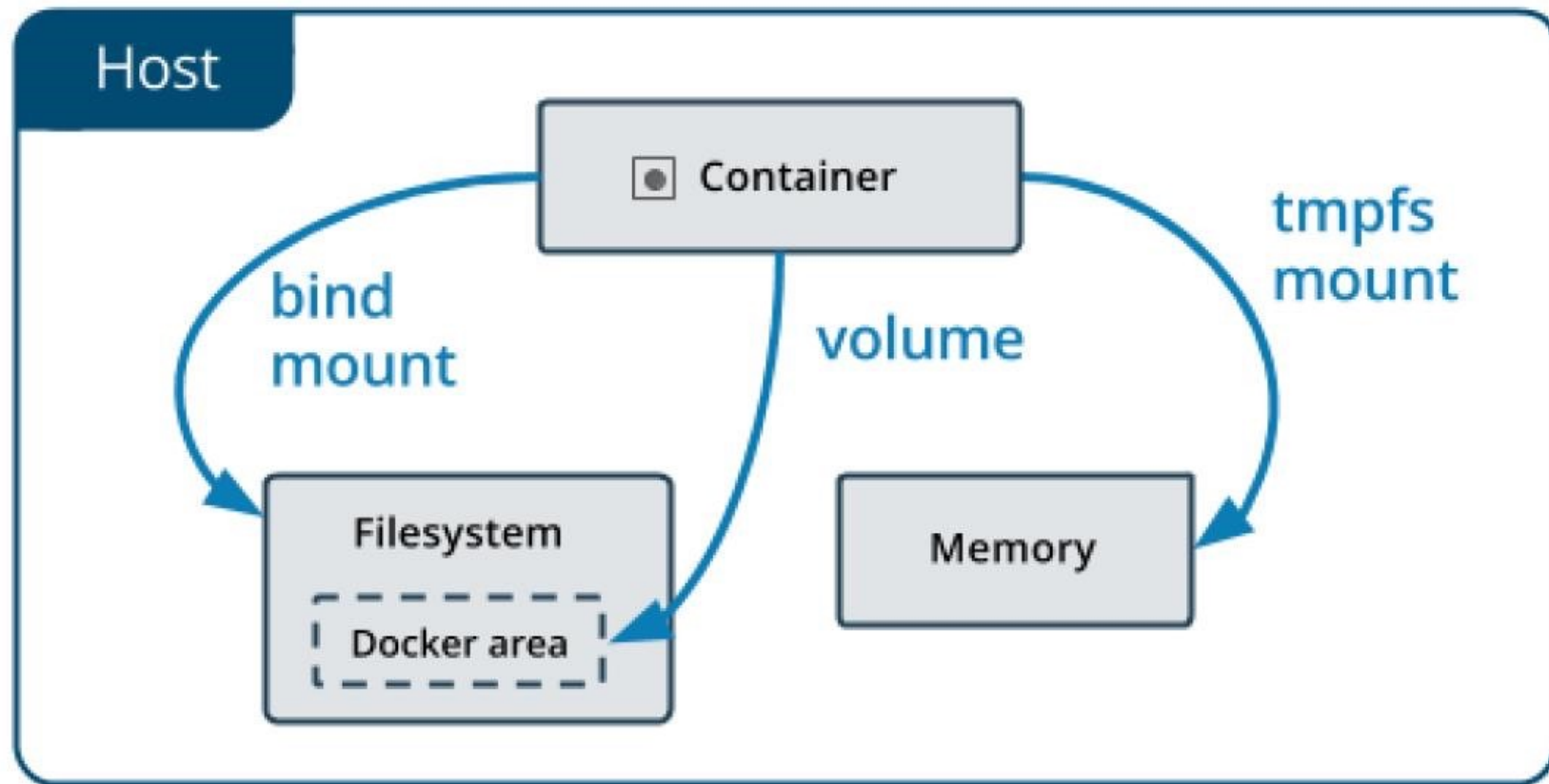
- Um servidor que habitualmente é do tipo *Daemon*, programa que executa um processo em *background*, daí o uso do comando **dockerd**;
- Uma REST API que especifica interfaces utilizadas para comunicar com o processo *Daemon* e indicar como este se deve comportar;
- E uma CLI, utilizada habitualmente com os comandos **docker**.



# Workflow Container



# Workflow informação



# Principais comandos

Command	Description
<b>docker pull</b>	Pull an image or a repositor from the registry
<b>docker run</b>	Run a command in a new container
<b>docker ps</b>	List the containers
<b>docker container</b>	Manage containers
<b>docker images</b>	List images
<b>docker start</b>	Start one or more stopped containers
<b>docker stop</b>	Stop one or more running containers
<b>docker inspect</b>	Return low-level information on docker objects
<b>docker rm</b>	Remove one or more containers
<b>docker rmi</b>	Remove one or more images
<b>docker logs</b>	Fetch logs from a container
<b>docker build</b>	Build na image from a Dockerfile
<b>docker tag</b>	Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

# Referências

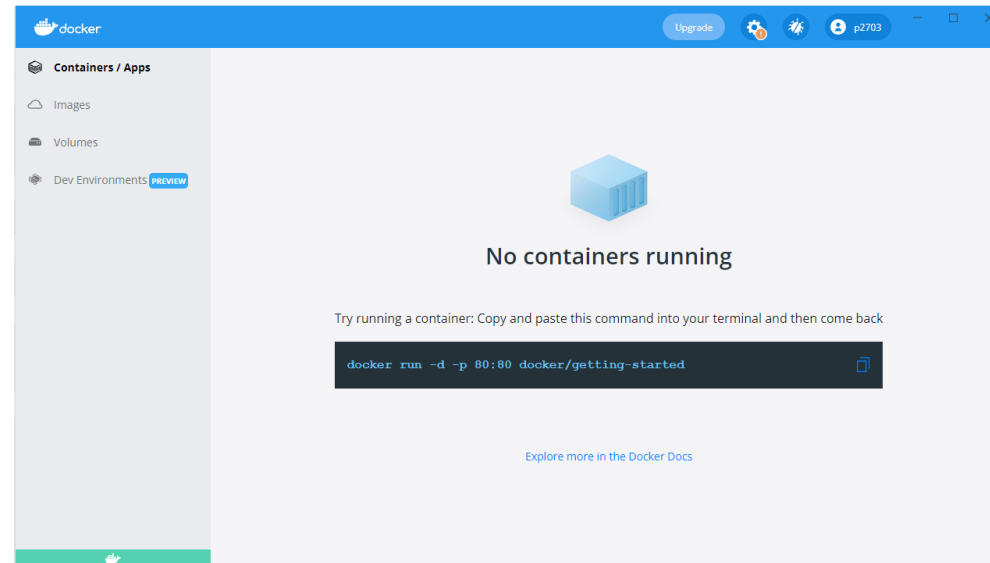
Docker get started	<a href="https://docs.docker.com/get-started/">https://docs.docker.com/get-started/</a>
Docker registry	<a href="https://docs.docker.com/registry/">https://docs.docker.com/registry/</a>
Docker file reference	<a href="https://docs.docker.com/engine/reference/builder/">https://docs.docker.com/engine/reference/builder/</a>
Docker compose	<a href="https://docs.docker.com/compose/">https://docs.docker.com/compose/</a>
Docker compose file	<a href="https://docs.docker.com/compose/compose-file/">https://docs.docker.com/compose/compose-file/</a>
Docker Swarm	<a href="https://docs.docker.com/engine/swarm/">https://docs.docker.com/engine/swarm/</a>
Container technologies overview	<a href="https://dzone.com/articles/container-technologies-overview">https://dzone.com/articles/container-technologies-overview</a>
Docker image repository	<a href="https://hub.docker.com/">https://hub.docker.com/</a>

# Docker: Instalar Desktop

<https://www.docker.com/products/docker-desktop/>

Válida para Windows e MacOS

O Docker-Desktop também instala ferramentas de *command line*



NOTA: A instalação de *Docker em ambiente Windows* requer a activação da componente *Windows System for Linux (WSL)*

<https://docs.microsoft.com/en-us/windows/wsl/>



# Docker:AWS Linux 2

1. Actualizar pacotes  
`sudo yum update -y`
2. Instalar *Docker Engine*  
`sudo amazon-linux-extras install docker`
3. Iniciar *Docker*  
`sudo service docker start`
4. Verificar estado do serviço e versão  
`docker info`

NOTA: As indicações referem-se a instalação em *Amazon Linux 2 (AWS)*  
O processo pode variar em função da implementação *Linux* utilizada  
Ex.: Ubuntu - <https://docs.docker.com/engine/install/ubuntu/>

# Docker: Container simples

- A instanciação de um *CONTAINER* requer uma *IMAGEM* e, opcionalmente, de *tag* com identificação de versão  
(ex.: `mysql:latest` instancia a última versão da *IMAGEM* de MySQL  
Se omitida a *tag* de versão, é utilizada a versão *latest*)  
A versão refere-se à imagem, não ao software base
- Consultar <https://hub.docker.com/> para todas as imagens oficiais para *Docker*
- Para instanciar um *CONTAINER* executa-se comando `RUN`, que pode ser acrescido de parâmetros de personalização  
Os parâmetros dependem da imagem e do ambiente  
(ex.: `-p` mapeia uma porta do *host* para a *IMAGEM* do *CONTAINER*)

# Docker: Container simples

Instanciar um CONTAINER com SGBD MySQL/MariaDB

```
docker run --name mysqldb  
            --env MARIADB_ROOT_PASSWORD=password  
            -p 3306:3306  
            -d mariadb:latest
```

Onde:

**--name** :: indica *tag* de identificação do *CONTAINER*

**--env** :: define variáveis de ambiente para o *CONTAINER*

Se houver mais do que uma variável, o parâmetro tem que ser repetido

Também se pode utiliza **-e**

**-p** :: mapeia um porto ip do *host* para o *CONTAINER*

**-d** :: indica que o *CONTAINER* será executado em *background*

# Docker: Container simples

Após instanciação, pode-se verificar o *CONTAINER* com comando **ps**

Ex.:

1. Instanciar *CONTAINER*: RUN

```
docker run --name mysqldb --env MARIADB_ROOT_PASSWORD=password -p 3306:3306 -d mariadb:latest
```

2. Verificar: PS (-a mostra contentores parados)

```
docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
149327880c95	mariadb:latest	"docker-entrypoint.s..."	6 seconds ago	Up 5 seconds	0.0.0.0:3306->3306/tcp	mysqldb

3. Parar *CONTAINER*: STOP (status indica *Exited*)

```
docker stop mysqldb
```

mysqldb

```
docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
149327880c95	mariadb:latest	"docker-entrypoint.s..."	4 minutes ago	Exited (0) 3 seconds ago		mysqldb

4. Retomar *CONTAINER*: START (status volta a *Up*)

```
docker start mysqldb
```

mysqldb

```
docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
149327880c95	mariadb:latest	"docker-entrypoint.s..."	7 minutes ago	Up 3 seconds	0.0.0.0:3306->3306/tcp	mysqldb

# Docker: Personalização

O *Docker* permite construir imagens personalizadas

Necessário indicar:

- Imagem base
- Ficheiro, em linguagem própria, com customizações

Tem que ter nome *Dockerfile*

- Pode incluir ficheiros/recursos adicionais a serem colocados na imagem

Para criar a imagem:

- Comando ***BUILD***
- *Dockerfile* e outros recursos têm que estar numa estrutura de pasta indicado como parâmetro do comando
- É gerada uma imagem, não um contentor

# Docker: Personalização

Construir imagem baseada em MariaDB com base de dados definida e p

## 1. *Dockerfile*

```
1 FROM mariadb:latest ← Imagem base
2
3 ADD ./sql/script.sql /docker-entrypoint-initdb.d { copia o ficheiro script.sql da pasta sql
4                                     para ficheiro initdb do CONTAINER
5                                     Este ficheiro é executado no BUILD
6
7 ENV MYSQL_ROOT_PASSWORD admin } Variáveis de ambiente para o CONTAINER
8 ENV MYSQL_DATABASE LD01INFO01 Substitui o parâmetro --env em RUN
9 ENV MYSQL_USER admin
10 ENV MYSQL_PASSWORD admin
11
12 RUN apt-get update && apt-get -y install vim { executa comandos dentro do CONTAINER
13                                     no exemplo, actualiza o SO
14
15 EXPOSE 3306 ← publica a porta 3306 para o host
16
17 CMD ["mysqld"] ← executa o comando MYSQLD dentro do CONTAINER
```

## 2. Comando *BUILD*

*Docker build .*

Onde “.” indica que *dockerfile* e a pasta /sql estão na pasta onde é executado

O comando

# Docker: Personalização

Após definição da IMAGEM, pode-se verificar *com comando **images***

Ex.:

1. Criar imagem personalizada: **BUILD** (-t atribui *tag* de nome à imagem)

```
docker build -t db .  
[+] Building 0.9s (8/8) FINISHED  
=> [internal] load build definition from Dockerfile  
=> => transferring dockerfile: 32B  
=> [internal] load .dockerignore  
=> => transferring context: 2B  
=> [internal] load metadata for docker.io/library/mariadb:latest  
=> [internal] load build context  
=> => transferring context: 70B  
=> [1/3] FROM docker.io/library/mariadb:latest@sha256:ca04948aca834  
=> CACHED [2/3] ADD ./sql/HR_Dump_20210213.sql /docker-entrypoint-  
=> CACHED [3/3] RUN apt-get update && apt-get -y install vim  
=> exporting to image  
=> => exporting layers  
=> => writing image sha256:606dfcad370b35ef746e85a45b052c1e8d27038  
=> => naming to docker.io/library/db
```

2. Verificar: IMAGES

docker images				
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
db	latest	606dfcad370b	25 hours ago	478MB

# Docker: Personalização

A instanciação de *CONTAINER* com *IMAGEM* CUSTOMIZADA é em tudo igual a imagens simples, só há que ter atenção ao nome da imagem e ser coerente com os parâmetros do *dockerfile*:

3. Instanciar *CONTAINER*: RUN

```
docker run --name mycustimdb -p 3306:3306 -d db
```

4. Verificar: PS

```
docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ce6caa75af94	db	"docker-entrypoint.s..."	34 seconds ago	Up 32 seconds	0.0.0.0:3306->3306/tcp	mycustimdb

5. Entrar no *CONTAINER*: EXEC (a ver em mais detalhe posteriormente...)

```
docker exec -it mycustimdb sh

# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 10.9.2-MariaDB-1:10.9.2+maria~ubu2204 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> |
```





Obrigado.