



4^a EDIÇÃO

Banco de dados

luiiza
<CODE>

Ementa:

1. A diferença entre SGBD e banco de dados
2. O que é banco de dados relacional
3. Instalando e configurando o PostgreSQL (VSCode)
4. Modelagem Banco de Dados Relacional
5. Relacionamento entre tabelas
6. SQL Nativo (DDL, DML, DQL)
7. Manipulação de dados SQL

1. A diferença entre SGBD e banco de dados

1.1 Banco de dados

Banco de dados ou base de dados pode simplesmente ser definida como um local – físico ou virtual – onde se **armazenam** dados de forma **organizada** e **indexada**. Percebiam que não estamos nem na parte de informática e sim na parte física e analógica.

Com isso em mente, podemos citar como exemplo de banco de dados, o bloco de notas, o documento no word, uma planilha, um caderno e até mesmo um baú. A internet também pode ser considerada um banco de dados.

Um banco de dados tem as seguintes propriedades implícitas:

- representa algum aspecto do mundo real;
- é uma coleção de dados logicamente coerente;
- é projetado, construído e populado para um propósito específico.

1. A diferença entre SGBD e banco de dados

1.2 Sistema de Gerenciamento de Banco de Dados (SGBD)

Um SGBD é um software responsável por tornar o banco de dados gerenciável permitir que ele seja manipulado, ele ainda não é o software que vai ao usuário final e sim o que o DBA ou Database Admin irá utilizar. Sua principal função é facilitar a interface com o banco e do DBA. Um SGBD é um **software de propósito geral** que facilita o processo de definir, construir, manipular e compartilhar bancos de dados entre vários usuários e aplicações.

Segue uma lista de **SGBDs** mais utilizados:

- Oracle
- MySQL
- SQL Server
- PostgreSQL
- MongoDB

2. O que é banco de dados relacional

2.1 Banco de dados relacional

Um banco de dados relacional é uma **coleção de itens** de dados com **relacionamentos** predefinidos entre si. Esses itens são organizados como um conjunto de **tabelas** com colunas e linhas.

As tabelas são usadas para reter informações sobre os objetos a serem representados no banco de dados. Cada coluna da tabela retém um determinado **tipo de dado** e um campo armazena o **valor** em si de um atributo.

As linhas na tabela representam uma coleção de valores relacionados de um objeto ou de uma **entidade**. Cada linha em uma tabela pode ser marcada com um único identificador chamado de **chave principal**. Já as linhas entre as várias tabelas podem ser associadas usando **chaves estrangeiras**.

Esses dados podem ser acessados de várias maneiras diferentes sem reorganizar as próprias tabelas do banco de dados.

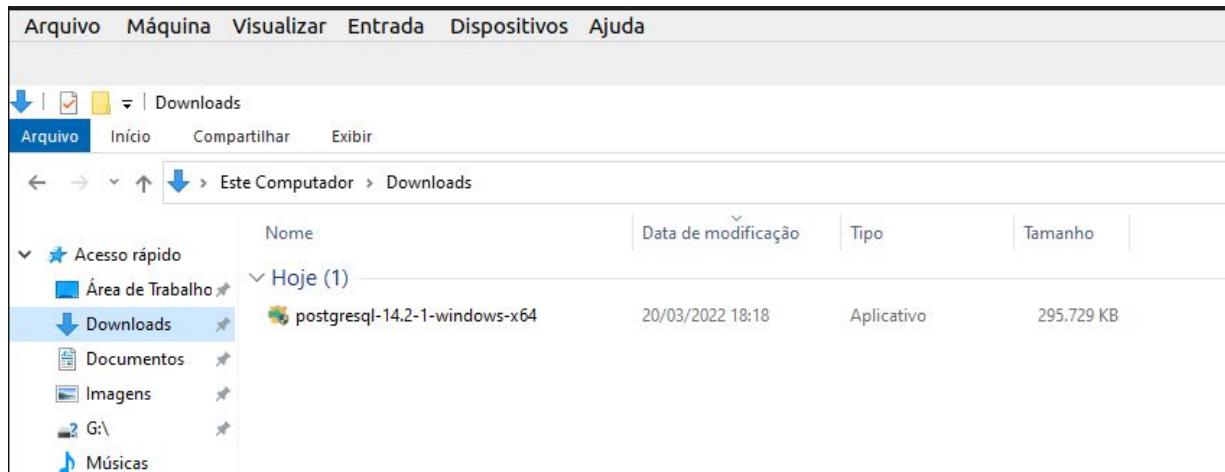
3. Instalando e configurando o PostgreSQL e DBeaver

3.1 Instalação do PostgreSQL no Windows

Realize o download do instalador no link

<https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>

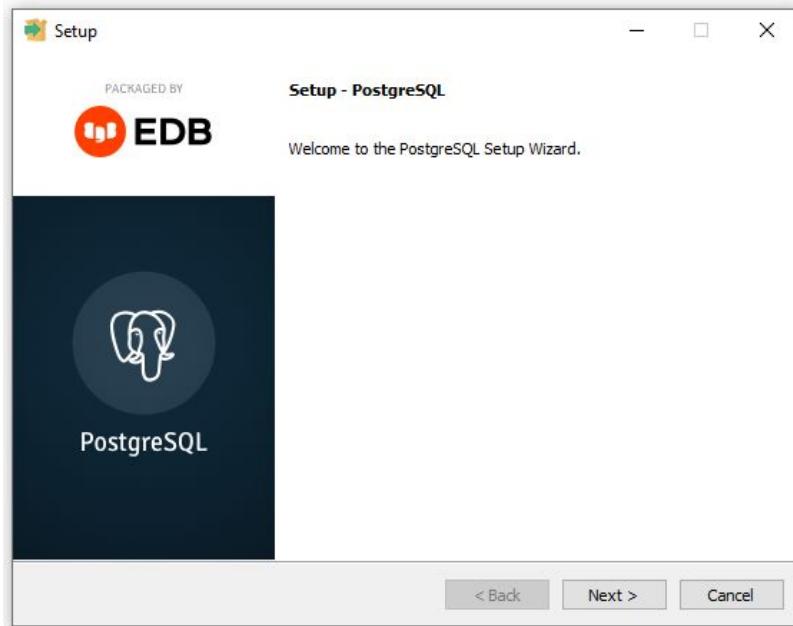
Após a finalização do download, execute o instalador:



3. Instalando e configurando o PostgreSQL e DBeaver

3.1 Instalação do PostgreSQL no Windows

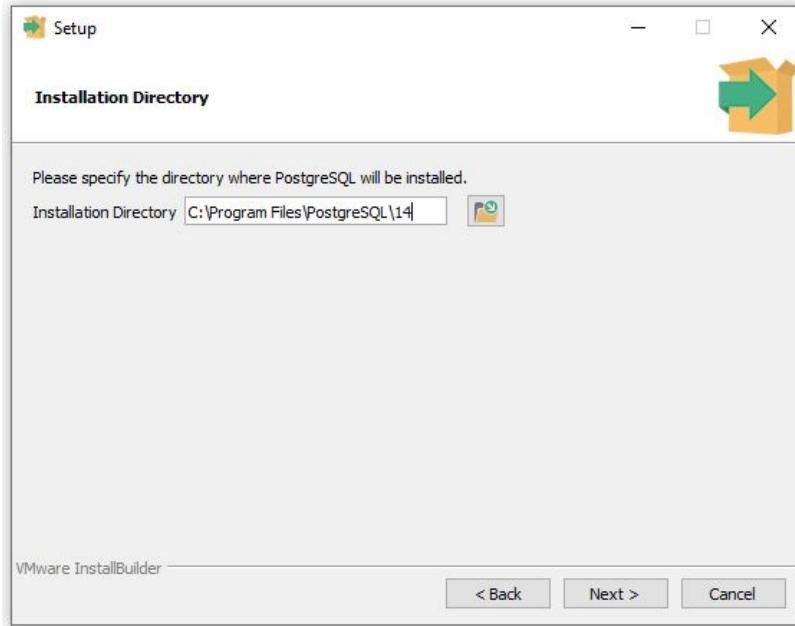
Siga para o próximo passo através da opção “Next”:



3. Instalando e configurando o PostgreSQL e DBeaver

3.1 Instalação do PostgreSQL no Windows

Siga para o próximo passo através da opção “Next”:

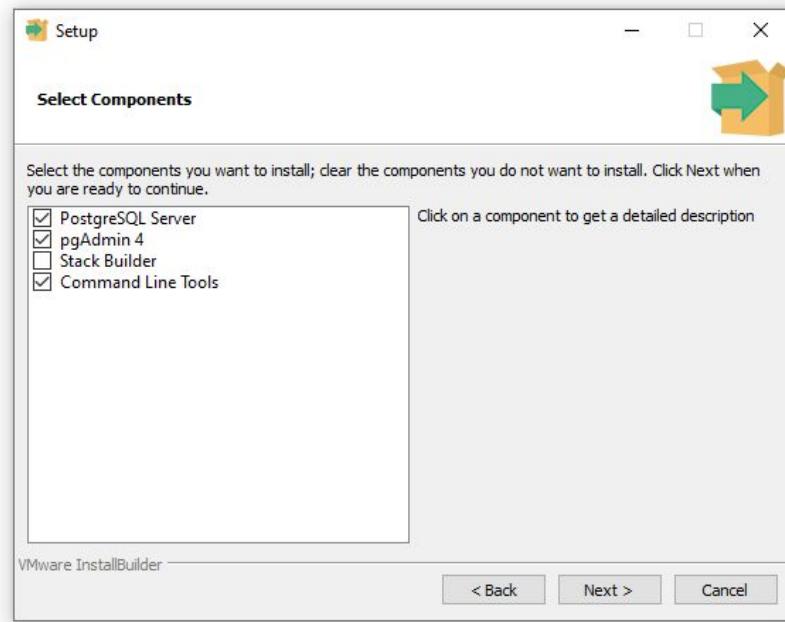


3. Instalando e configurando o PostgreSQL e DBeaver

3.1 Instalação do PostgreSQL no Windows

Desmarque o componente “Stack Builder” que não será necessário.

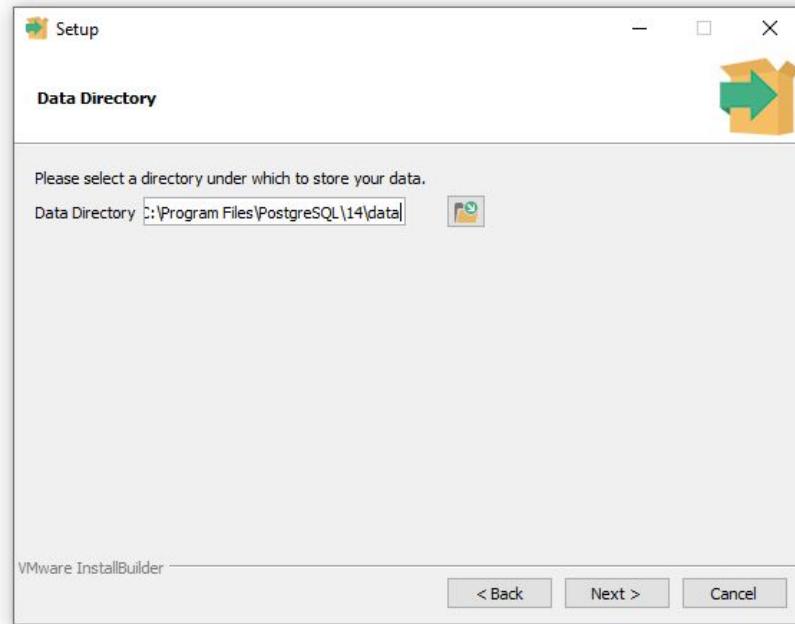
Siga para o próximo passo através da opção “Next”:



3. Instalando e configurando o PostgreSQL e DBeaver

3.1 Instalação do PostgreSQL no Windows

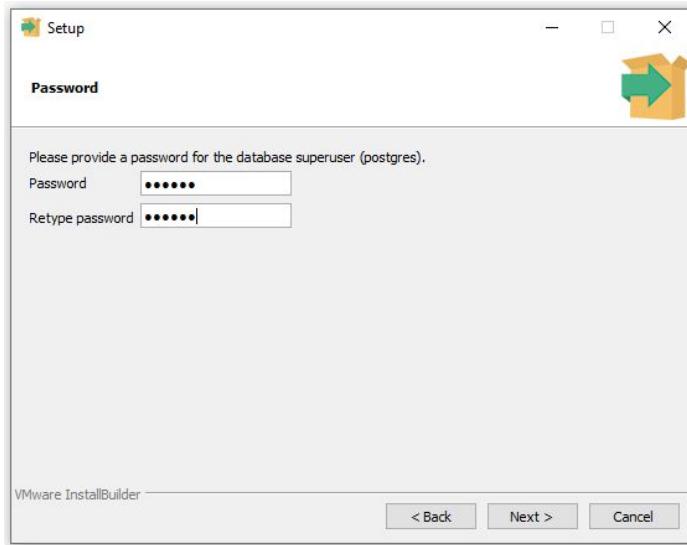
Nesta etapa podemos indicar o diretório onde será armazenado os dados. No exemplo abaixo iremos manter o caminho sugerido pelo instalador. Siga para o próximo passo através da opção “Next”:



3. Instalando e configurando o PostgreSQL e DBeaver

3.1 Instalação do PostgreSQL no Windows

Aqui definimos uma senha de administrador do banco de dados. É extremamente importante manter essa senha de uma forma segura. Siga para o próximo passo através da opção “Next”:

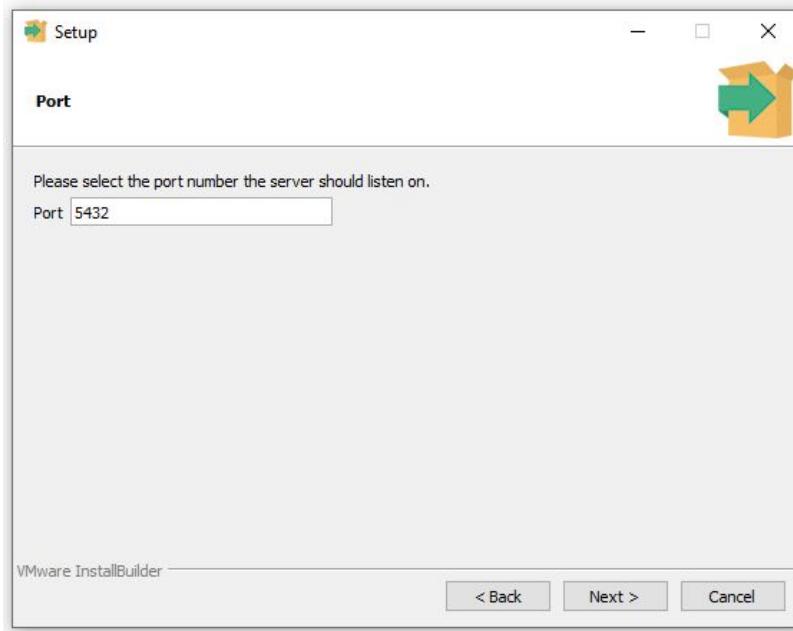


3. Instalando e configurando o PostgreSQL e DBeaver

3.1 Instalação do PostgreSQL no Windows

Neste momento é configurado a porta que o servidor irá utilizar. Iremos manter a configuração padrão.

Siga para o próximo passo através da opção “Next”:

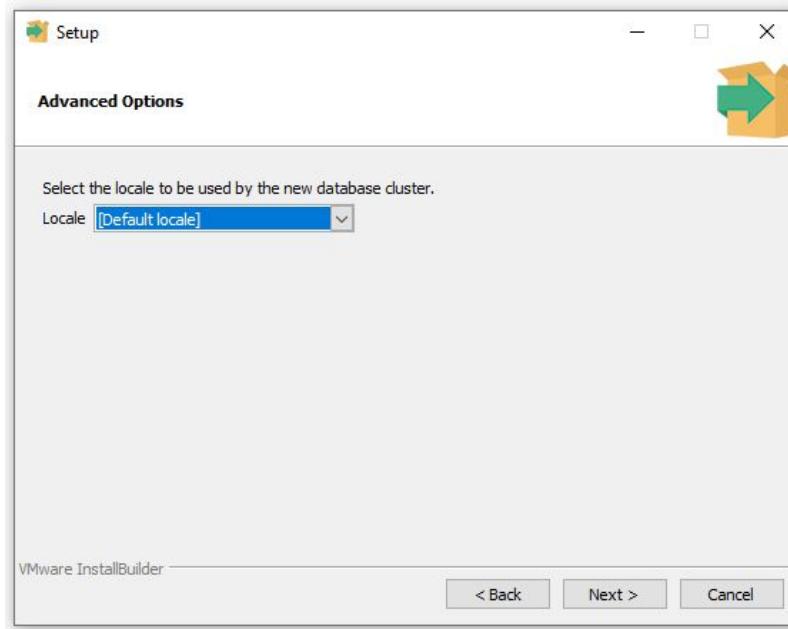


3. Instalando e configurando o PostgreSQL e DBeaver

3.1 Instalação do PostgreSQL no Windows

Iremos utilizar a configuração de local default.

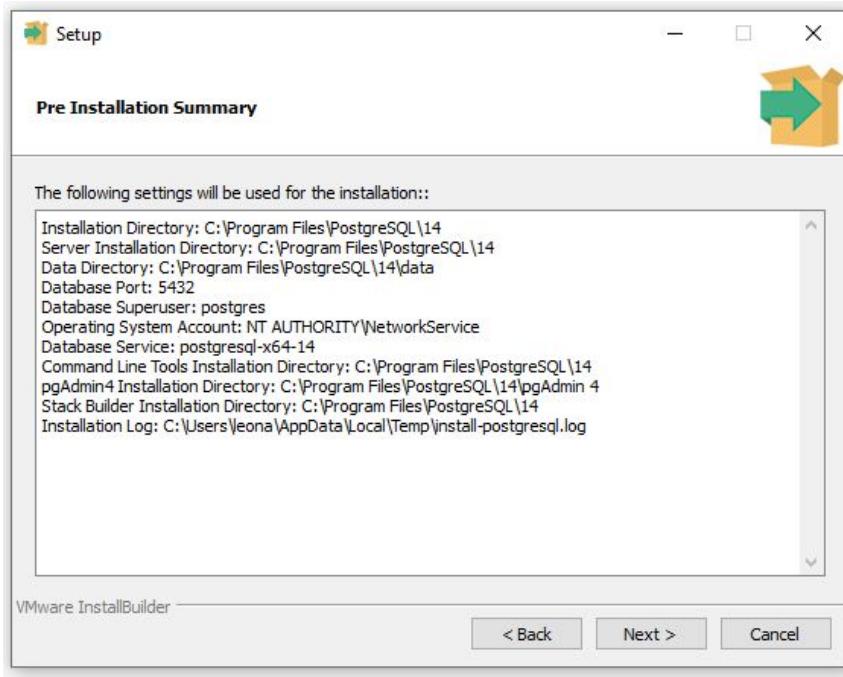
Siga para o próximo passo através da opção “Next”:



3. Instalando e configurando o PostgreSQL e DBeaver

3.1 Instalação do PostgreSQL no Windows

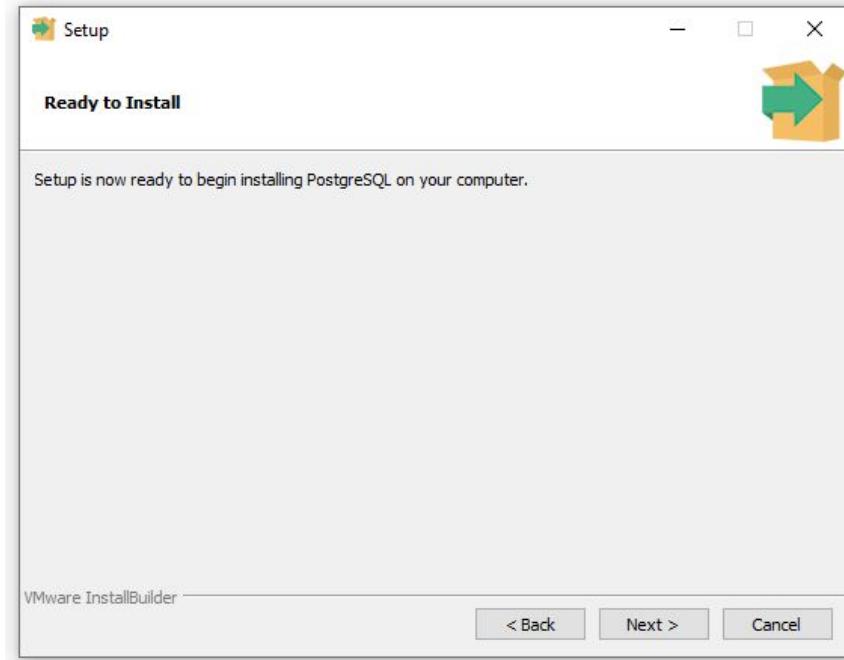
Siga para o próximo passo através da opção “Next”:



3. Instalando e configurando o PostgreSQL e DBeaver

3.1 Instalação do PostgreSQL no Windows

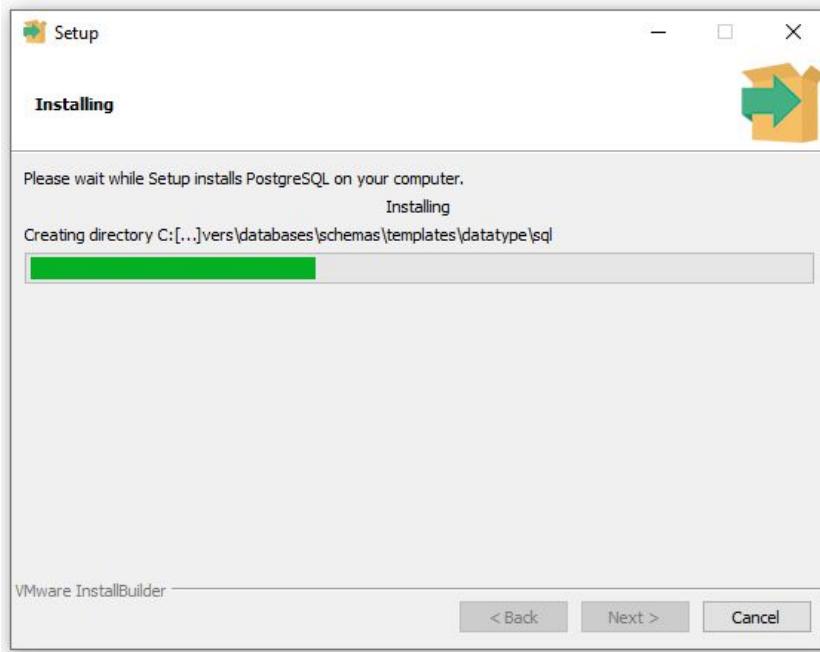
Siga para o próximo passo através da opção “Next”:



3. Instalando e configurando o PostgreSQL e DBeaver

3.1 Instalação do PostgreSQL no Windows

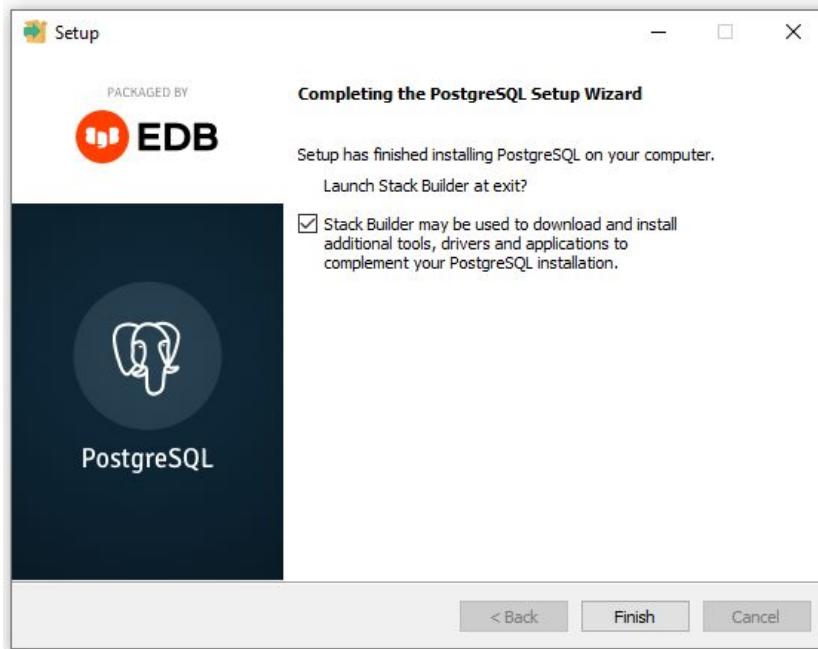
Aguarde a instalação finalizar.



3. Instalando e configurando o PostgreSQL e DBeaver

3.1 Instalação do PostgreSQL no Windows

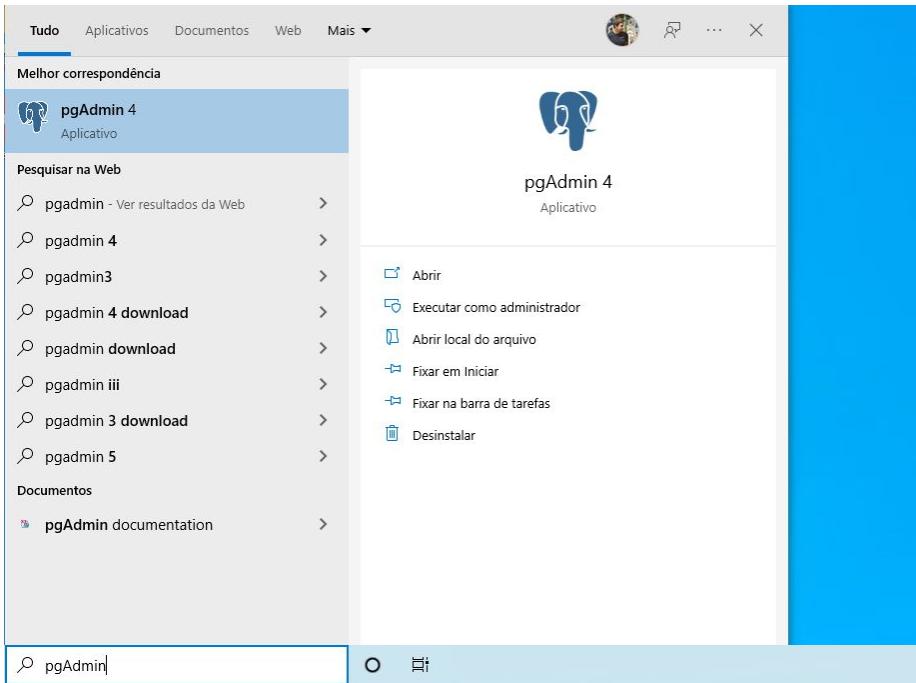
Finalize a instalação.



3. Instalando e configurando o PostgreSQL e DBeaver

3.2 Criação do banco de dados

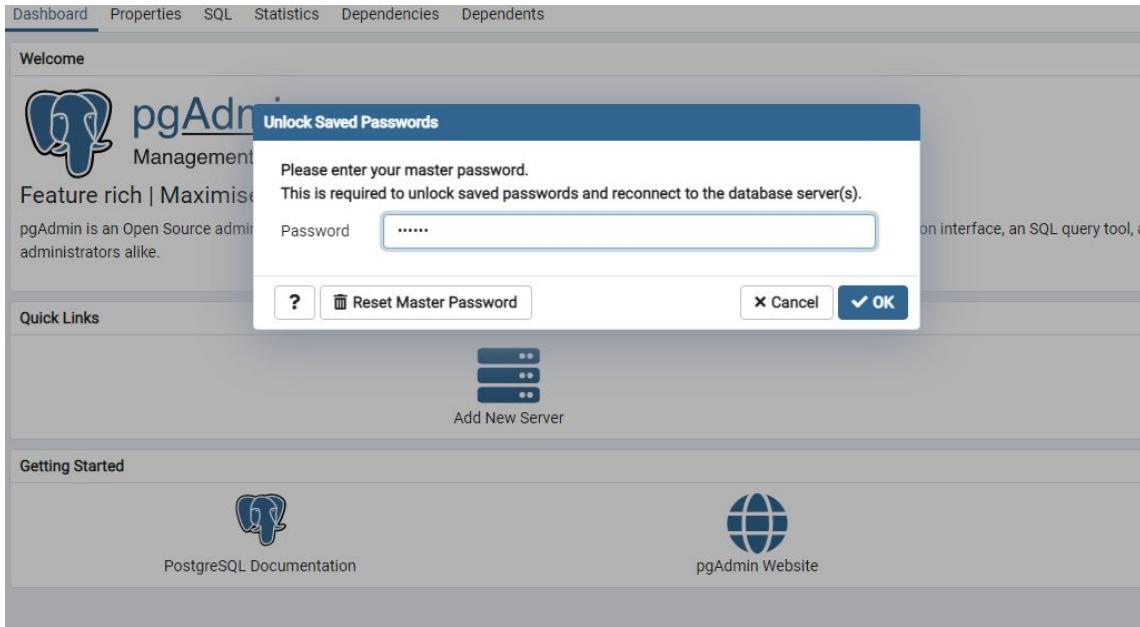
Iremos utilizar o pgAdmin 4 (que também foi instalado no pacote) para realizar a criação do banco de dados:



3. Instalando e configurando o PostgreSQL e DBeaver

3.2 Criação do banco de dados

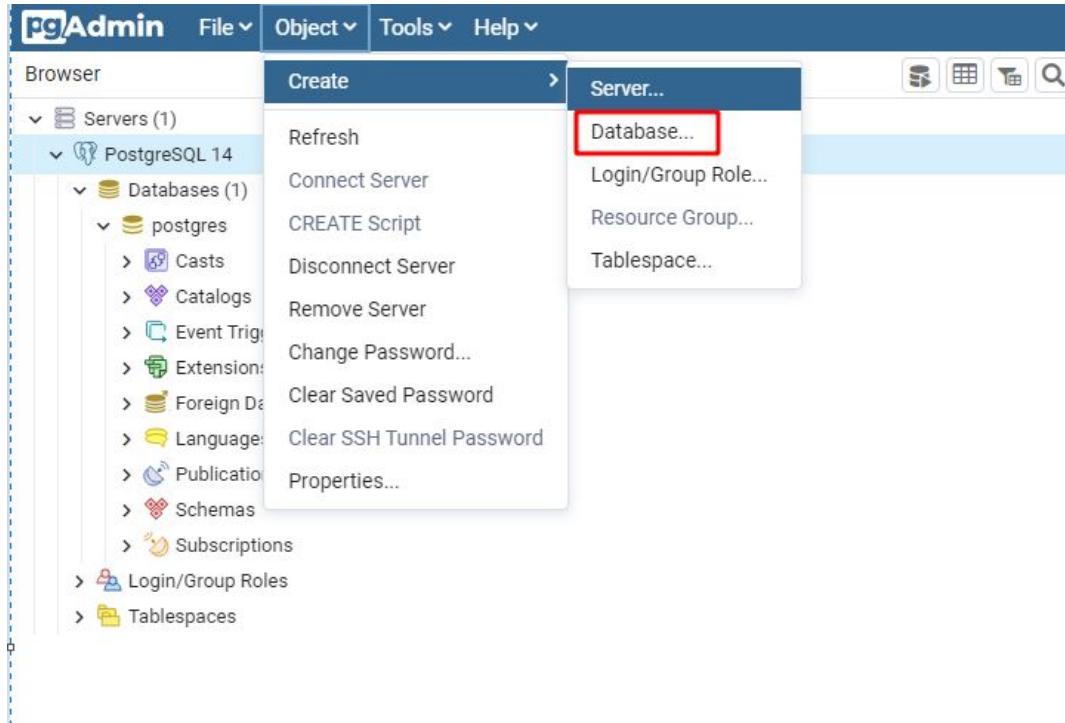
Insira a senha de administrador do banco de dados que foi configurada na instalação:



3. Instalando e configurando o PostgreSQL e DBeaver

3.2 Criação do banco de dados

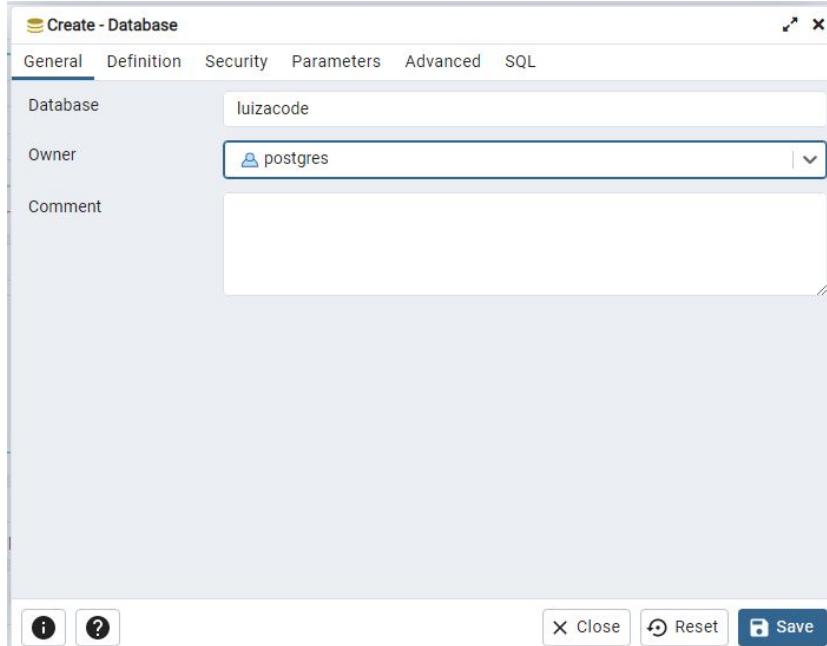
Selecione a opção Object -> Create -> Database...



3. Instalando e configurando o PostgreSQL e DBeaver

3.2 Criação do banco de dados

Defina o nome do banco de dados como **luizacode** e clique em “Save”.



3. Instalando e configurando o PostgreSQL e DBeaver

3.3 Instalação da IDE DBeaver

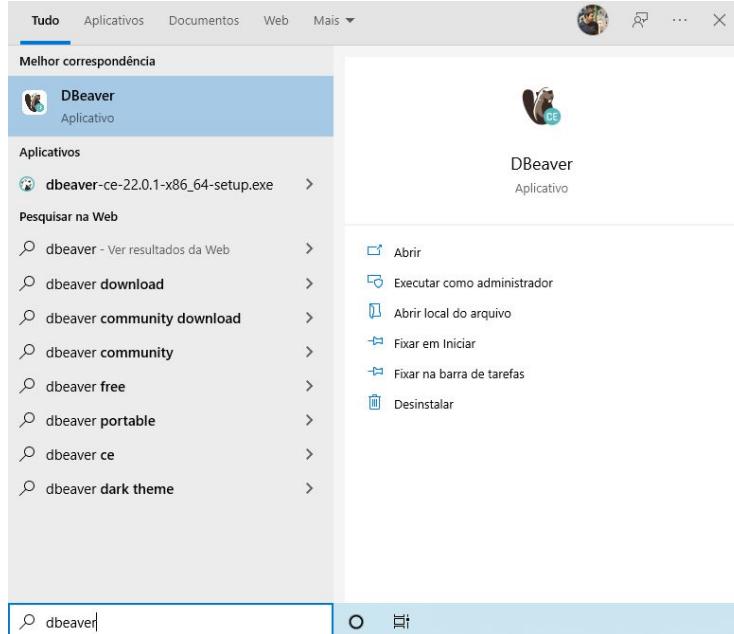
O DBeaver é um programa multiplataforma, que tem por objetivo conectar e manipular vários tipos de banco de dados. Realize o download através do link <https://dbeaver.io/download>.



3. Instalando e configurando o PostgreSQL e DBeaver

3.3 Instalação da IDE DBeaver

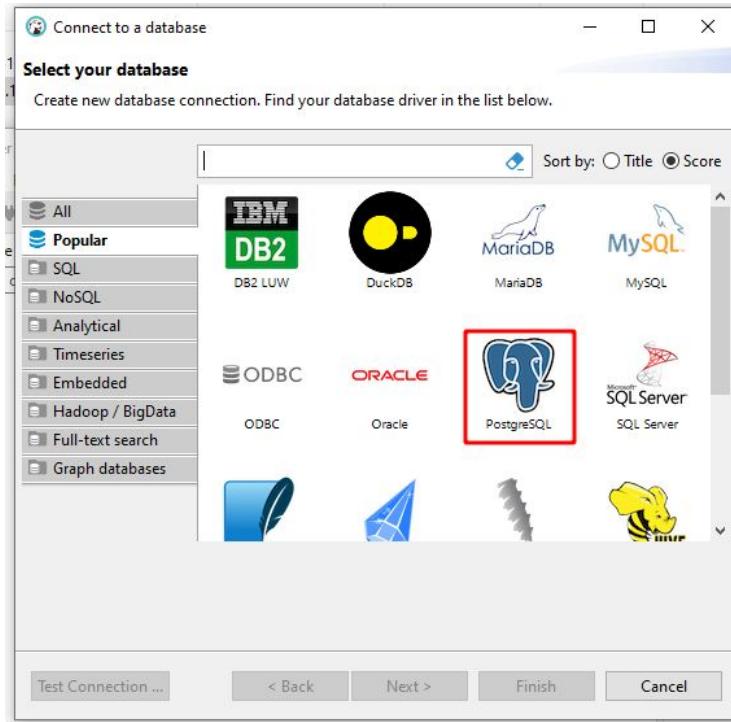
Inicie o DBeaver:



3. Instalando e configurando o PostgreSQL e DBeaver

3.3 Instalação da IDE DBeaver

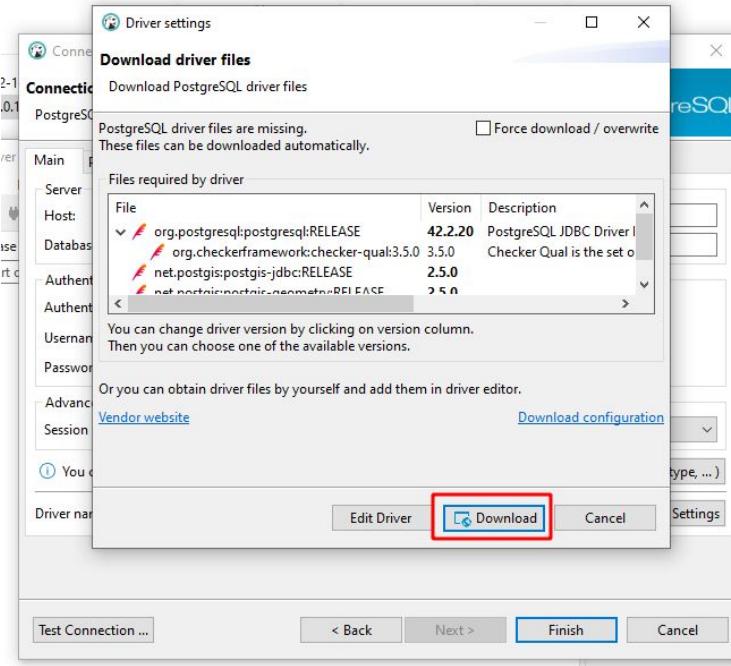
Selecione o opção PostgreSQL:



3. Instalando e configurando o PostgreSQL e DBeaver

3.3 Instalação da IDE DBeaver

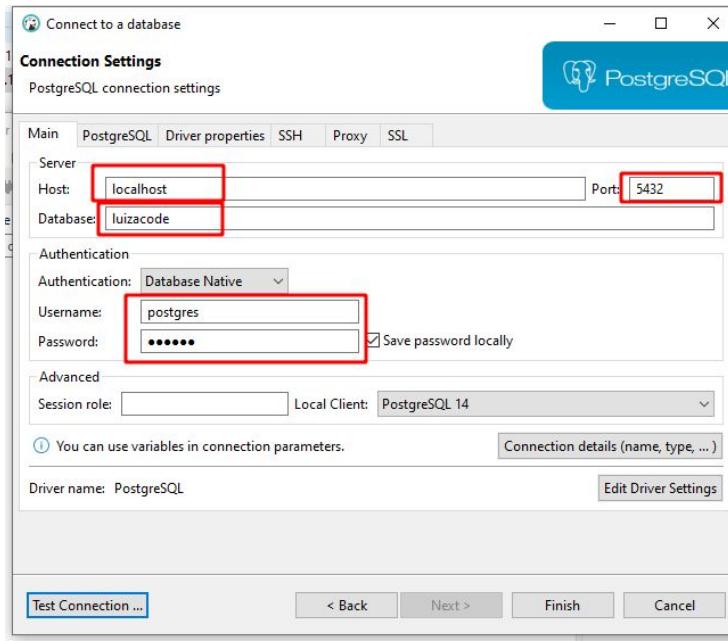
A próxima etapa é realizar o download dos drivers necessários para realizar a conexão com o banco de dados. O DBeaver automaticamente sugere quais devem ser baixados:



3. Instalando e configurando o PostgreSQL e DBeaver

3.3 Instalação da IDE DBeaver

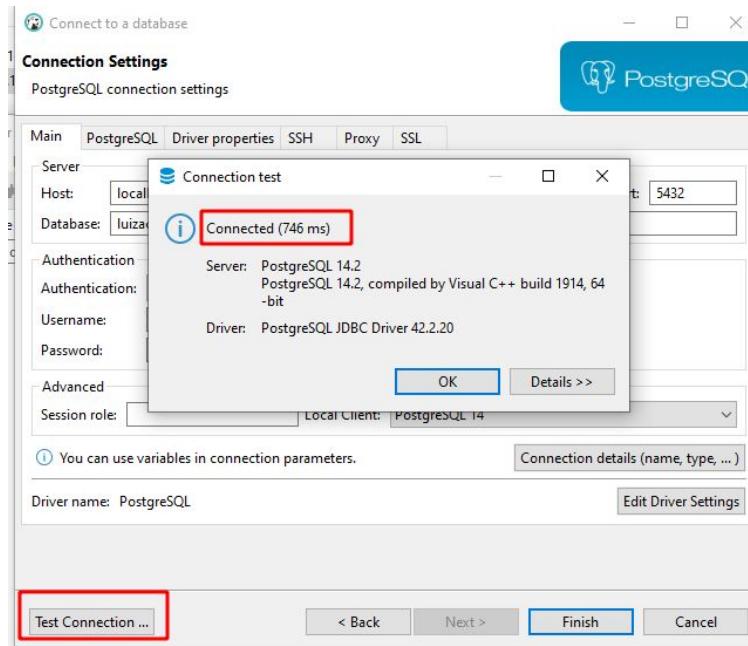
Agora inserimos as informações de conexão com o banco de dados. Aqui novamente utilizamos a senha definida na instalação.



3. Instalando e configurando o PostgreSQL e DBeaver

3.3 Instalação da IDE DBeaver

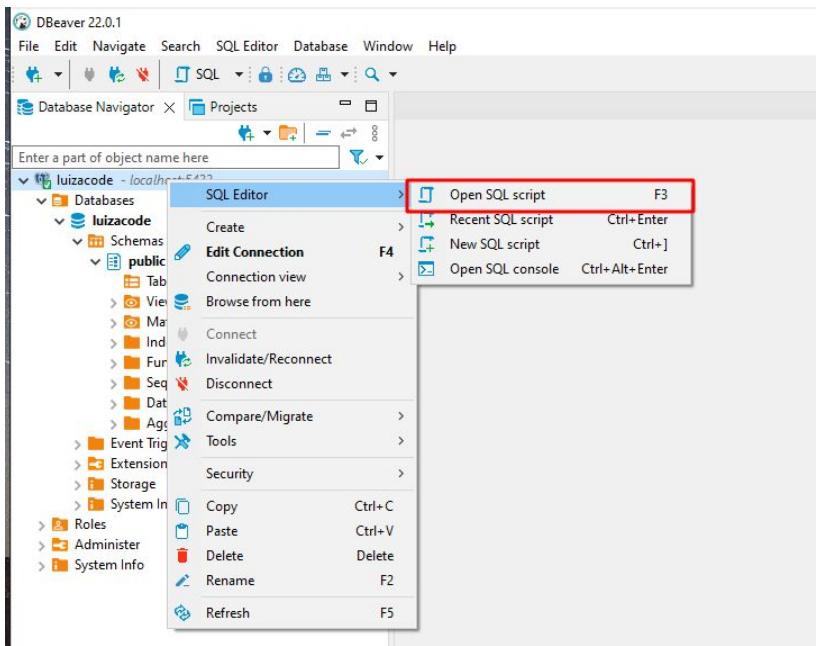
É aconselhável testar a conexão para confirmar se todos os dados foram inseridos corretamente.



3. Instalando e configurando o PostgreSQL e DBeaver

3.3 Instalação da IDE DBeaver

Agora que conseguimos conectar, iremos abrir um editor onde será possível realizar as operações de criação e manipulação do banco de dados.



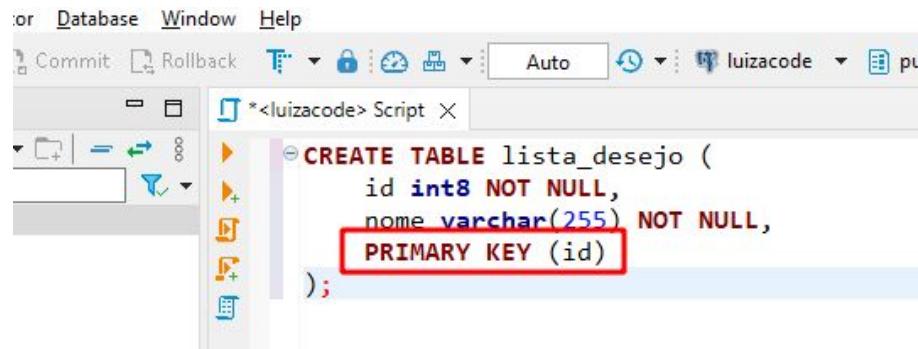
4. Modelagem do banco de dados

4.1 Primary key

A **PRIMARY KEY** é uma restrição que identifica **exclusivamente** cada registro em uma tabela.

As chaves primárias devem conter valores **únicos** e não podem conter valores **NULL**.

Uma tabela pode ter apenas **UMA** chave primária; e na tabela, essa chave primária pode consistir em uma ou várias colunas (campos).



```
File Database Window Help
Commit Rollback Auto luizacode pu
* <luizacode> Script X
CREATE TABLE lista_desejo (
    id int8 NOT NULL,
    nome varchar(255) NOT NULL,
    PRIMARY KEY (id)
);
```

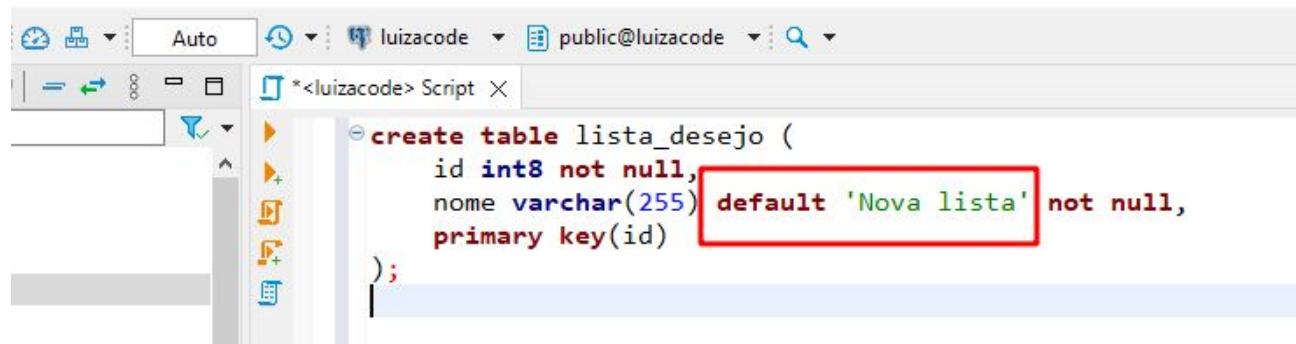
The screenshot shows the MySQL Workbench interface. The main window displays a SQL script titled '<luizacode> Script X'. The script contains a 'CREATE TABLE' statement for a table named 'lista_desejo'. The 'id' column is defined as 'int8 NOT NULL', and the 'PRIMARY KEY (id)' constraint is applied to it. A red rectangular box highlights the 'PRIMARY KEY (id)' line. The interface includes standard database management tools like 'Commit', 'Rollback', and connection dropdowns at the top.

4. Modelagem do banco de dados

4.2 Default

A constraint **DEFAULT** é utilizada para inserirmos um valor padrão em uma coluna. Esse valor padrão é inserido **automaticamente** nos registros, se nenhum valor for especificado para a coluna em questão.

Abaixo temos uma sintaxe para criar valor um padrão “Nova lista” na coluna **nome**:



The screenshot shows the pgAdmin 4 interface with a database connection to 'luizacode' and a schema of 'public@luizacode'. A script editor window titled '*<luizacode> Script X*' contains the following SQL code:

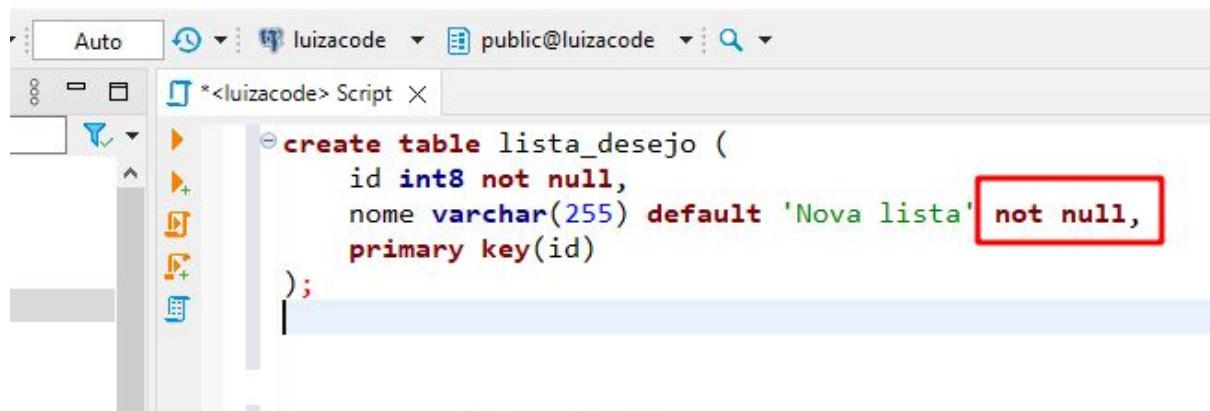
```
create table lista_desejo (
    id int8 not null,
    nome varchar(255) default 'Nova lista' not null,
    primary key(id)
);
```

The line 'default 'Nova lista' not null,' is highlighted with a red rectangular box.

4. Modelagem do banco de dados

4.3 Not null

A restrição **NOT NULL** garante que uma coluna não admite valores **NULL**. Isto significa que será abortada uma operação de INSERT ou UPDATE que coloque um valor **NULL** nessa coluna. A sua utilização é útil sempre que as regras de negócio obriguem ao preenchimento de um campo.



The screenshot shows a MySQL Workbench interface with the following details:

- Session: Auto
- Server: luizacode
- Database: public@luizacode
- Script tab: *<luizacode> Script
- Table creation query:

```
create table lista_desejo (
    id int8 not null,
    nome varchar(255) default 'Nova lista' not null,
    primary key(id)
);
```

The line "not null" in the "nome" column definition is highlighted with a red rectangular box.

4. Modelagem do banco de dados

4.4 Tipos de dados

O PostgreSQL tem um rico conjunto de tipos de dados nativos disponíveis para os usuários. Além deles, é possível ainda adicionar novos tipos ao PostgreSQL usando o comando **CREATE TYPE**.

Iremos apresentar os principais:

Nome	Alias	Descrição
bigint	int8	inteiro de oito bytes
bigserial	serial8	inteiro de oito bytes de autoincremento
boolean	bool	lógica booleana (verdadeiro/falso)
character	char [(n)]	cadeia de caracteres de comprimento fixo
character varying	varchar [(n)]	cadeia de caracteres de comprimento variável
date		data do calendário (ano, mês, dia)
integer	int, int4	inteiro de quatro bytes assinado
numeric	decimal	numérico exato de precisão

4. Modelagem do banco de dados

4.4 Tipos de dados

O PostgreSQL tem um rico conjunto de tipos de dados nativos disponíveis para os usuários. Além deles, é possível ainda adicionar novos tipos ao PostgreSQL usando o comando **CREATE TYPE**.

Iremos apresentar os principais:

Nome	Alias	Descrição
bigint	int8	inteiro de oito bytes
bigserial	serial8	inteiro de oito bytes de autoincremento
boolean	bool	lógica booleana (verdadeiro/falso)
character	char [(n)]	cadeia de caracteres de comprimento fixo
character varying	varchar [(n)]	cadeia de caracteres de comprimento variável
date		data do calendário (ano, mês, dia)
integer	int, int4	inteiro de quatro bytes assinado
numeric	decimal	numérico exato de precisão

5. Relacionamento entre tabelas

5.1 Relacionamento um para um

São relacionamentos em que **uma ocorrência** de uma entidade em **A** está associada no máximo a uma ocorrência em uma entidade **B** e uma ocorrência na entidade **B** está associada no máximo a uma ocorrência na entidade **A**.

No exemplo abaixo podemos afirmar que uma lista de desejo pode somente conter uma categoria vinculada a ela.



5. Relacionamento entre tabelas

5.2 Relacionamento um para muitos

Um relacionamento 1:m ocorre com freqüência em situações de negócio. Às vezes ocorre em forma de árvore ou em **forma hierárquica**.

No exemplo abaixo, temos a seguinte representação: cada lista de desejo cadastrada possui **vários produtos** ligados a ela. O campo **lista_desejo_id** foi escolhido como chave estrangeira, que define de fato a ligação entre as duas tabelas. A representação ficaria assim:



6. SQL Nativo

6.1 Create table

A instrução **CREATE TABLE** é usada para criar uma nova tabela em um banco de dados.

Os parâmetros de **coluna** especificam os nomes das colunas da tabela.

O parâmetro **tipo** especifica o tipo de dados que a coluna pode conter (por exemplo, varchar, integer, date, etc.)



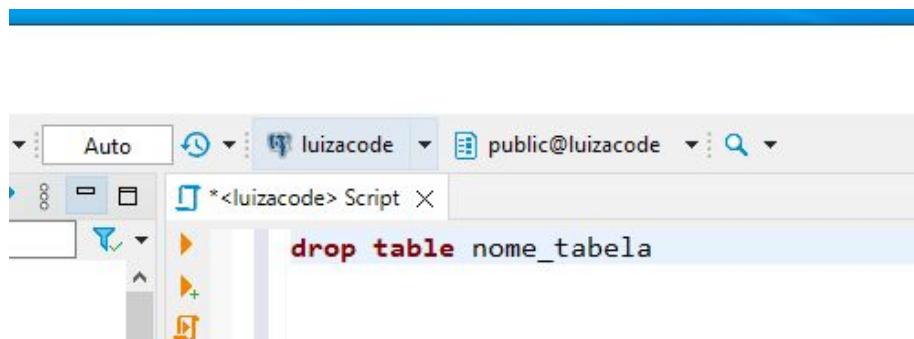
```
Auto luizacode public@luizacode *<luizacode> Script X
create table nome_tabela (
    campo tipo,
    campo tipo,
    ...
);
```

6. SQL Nativo

6.2 Drop table

A instrução **DROP TABLE** é usada para excluir uma tabela em um banco de dados.

A seguir a sintaxe da instrução:



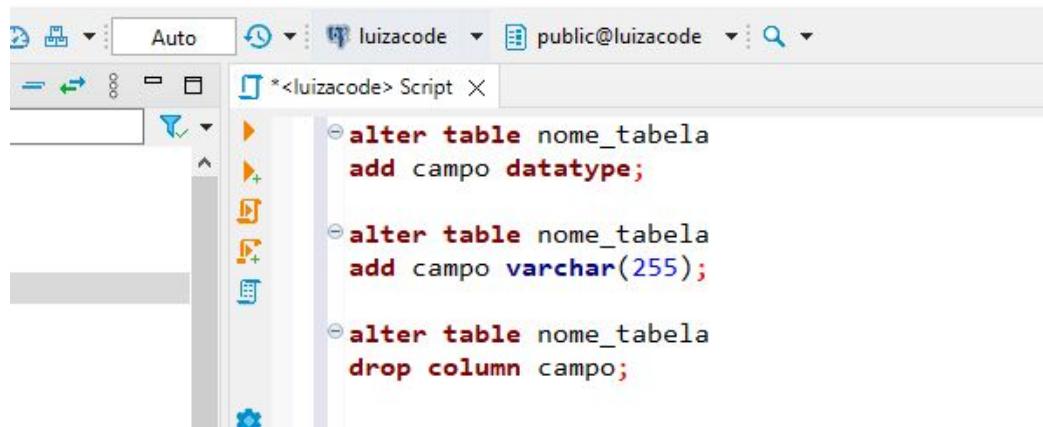
The screenshot shows a PostgreSQL pgAdmin interface. The top bar displays the connection information: 'Auto', 'luizacode', 'public@luizacode'. Below the bar, there's a toolbar with various icons. The main area is titled 'Script' and contains the SQL command: 'drop table nome_tabela'. The command is highlighted in red, indicating it is a keyword.

6. SQL Nativo

6.3 Alter table

A instrução **ALTER TABLE** é usada para adicionar, excluir ou modificar colunas em uma tabela existente.

A seguir a sintaxe da instrução de adição, exclusão ou alteração de campos:



```
alter table nome_tabela
add campo datatype;

alter table nome_tabela
add campo varchar(255);

alter table nome_tabela
drop column campo;
```

6. SQL Nativo

6.4 Insert into

A instrução **INSERT INTO** é usada para inserir novos registros em uma tabela.

Devemos especificar os **nomes** das colunas e os **valores** a serem inseridos:



The screenshot shows a MySQL Workbench interface. The title bar displays 'luizacode' and 'public@luizacode'. The main area contains a script editor with the following SQL code:

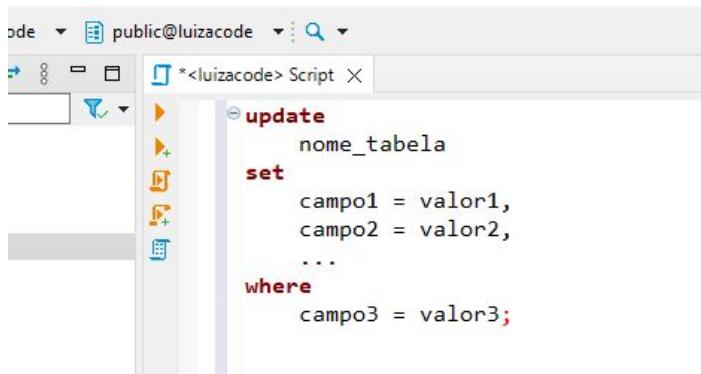
```
insert
into
nome_tabela (campo1,
campo2,
campo3)
values (valor1,
valor2,
valor3);
```

6. SQL Nativo

6.5 Update

A instrução **UPDATE** é usada para modificar os registros existentes em uma tabela.

É extremamente importante ter cuidado ao atualizar registros em uma tabela! Observe sempre a cláusula **WHERE** declaração. Ela especifica quais registros devem ser atualizados. Se você omitir essa cláusula, todos os registros da tabela serão atualizados!



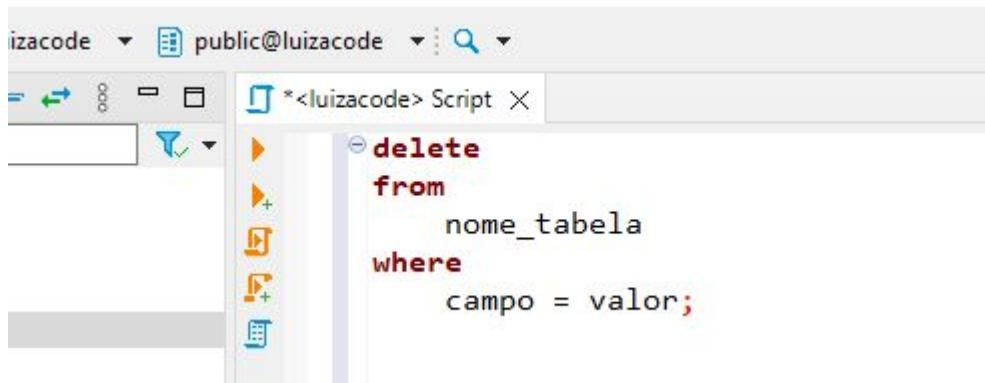
```
public@luizacode:~$ > update nome_tabela set campo1 = valor1, campo2 = valor2, ... where campo3 = valor3;
```

6. SQL Nativo

6.6 Delete

A instrução **DELETE** é usada para excluir registros existentes em uma tabela.

Para esse instrução também é muito importante ter cuidado com a cláusula **WHERE**. Caso contrário registros que não deveriam ser apagados podem ser impactados!



The screenshot shows a PostgreSQL pgAdmin interface. In the top left, it says 'izacode' and 'public@luizacode'. The main window has a toolbar with icons for connection, schema browser, and search. A vertical toolbar on the left has icons for file operations, a tree view, and a search bar. The central area is a 'Script X' tab containing the following SQL code:

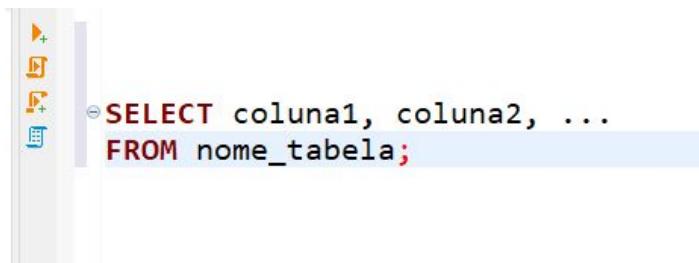
```
delete
from
    nome_tabela
where
    campo = valor;
```

7. Manipulação de dados SQL

7.1 Select

A cláusula **SELECT** é usada para selecionar dados de um banco de dados.

Os dados retornados são armazenados em uma tabela de resultados, chamada de conjunto de resultados.



The screenshot shows a code editor interface with a vertical toolbar on the left containing icons for file operations like new, open, save, and copy/paste. A code snippet is displayed in the main area:

```
④ SELECT coluna1, coluna2, ...
  FROM nome_tabela;
```

Aqui, coluna1, coluna2, ... são os nomes dos campos da tabela da qual você deseja selecionar os dados. Se você deseja selecionar todos os campos disponíveis na tabela, use a seguinte sintaxe:

7. Manipulação de dados SQL

7.2 Join

A cláusula **JOIN** é usada para combinar linhas de duas ou mais tabelas, com base em uma coluna relacionada entre elas.

Observe que a coluna **id** na tabela **lista_desejo** se refere ao "**lista_desejo_id**" na tabela "**produtos**". A relação entre as duas tabelas acima é a coluna "**lista_desejo_id**".

Em seguida, podemos criar a seguinte instrução SQL (que contém um INNER JOIN), que seleciona os registros que possuem valores correspondentes em **ambas as tabelas**:

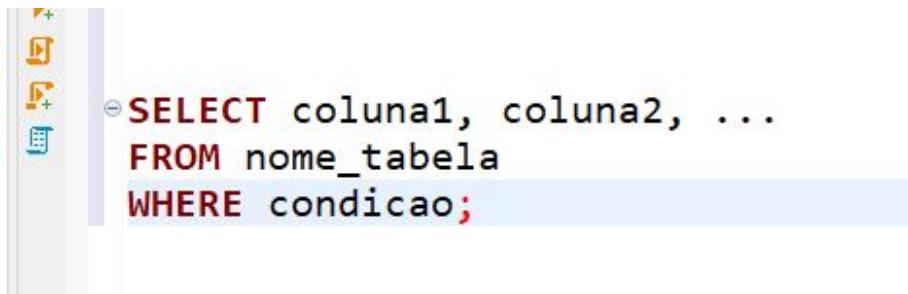
```
④ SELECT nome_tabela1.coluna1, nome_tabela2.coluna2
  FROM nome_tabela1
INNER JOIN nome_tabela2 ON nome_tabela1.coluna1 = nome_tabela2.coluna2;
```

7. Manipulação de dados SQL

7.3 Where

A cláusula **WHERE** é usada para filtrar registros.

Ele é usado para extrair apenas os registros que atendem a uma condição especificada.



```
④ SELECT coluna1, coluna2, ...
  FROM nome_tabela
 WHERE condicao;
```

Lembrando que a cláusula **WHERE** não é usada apenas em declarações **SELECT**, ela também é usada em **UPDATE**, **DELETE**, etc.!

7. Manipulação de dados SQL

7.4 Order by

A palavra-chave **ORDER BY** é usada para classificar o conjunto de resultados em ordem crescente ou decrescente.

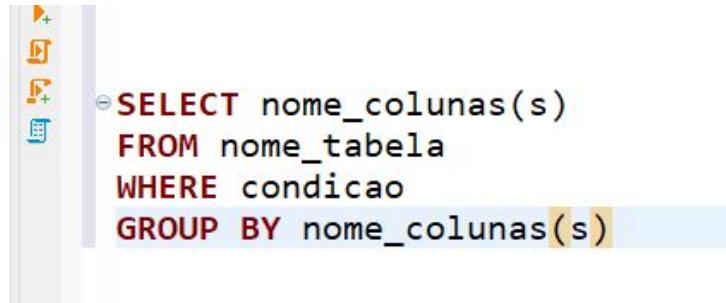
```
SELECT coluna1, coluna2, ...
FROM nome_tabela
ORDER BY coluna1, coluna2, ... ASC|DESC;
```

Elas classificam os registros em ordem crescente por padrão. Para classificar os registros em ordem decrescente, use a palavra-chave **DESC**.

7. Manipulação de dados SQL

7.5 Group by

A instrução **GROUP BY** agrupa as linhas que têm os mesmos valores em linhas de resumo.



```
④ SELECT nome_colunas(s)
  FROM nome_tabela
 WHERE condicao
 GROUP BY nome_colunas(s)
```

Ela é frequentemente usada com funções agregadas (**COUNT**, **MAX**, **MIN**, **SUM**, **AVG**) para agrupar o conjunto de resultados por uma ou mais colunas.

7. Manipulação de dados SQL

7.6 COUNT, AVG e SUM

A função **COUNT** retorna o número de linhas que corresponde a um critério especificado.

```
SELECT COUNT(nome_coluna)
  FROM nome_tabela
 WHERE condicao;
```

A função **AVG** retorna o valor médio de uma coluna numérica.

```
SELECT AVG(nome_coluna)
  FROM nome_tabela
 WHERE condicao;
```

A função **SUM** retorna a soma total de uma coluna numérica.

```
SELECT SUM(nome_coluna)
  FROM nome_tabela
 WHERE condicao;
```

7. Manipulação de dados SQL

7.7 MIN E MAX

A função **MIN** retorna o menor valor da coluna selecionada.



```
SELECT MIN(nome_coluna)
FROM nome_tabela
WHERE condicao;
```

A função **MAX** retorna o maior valor da coluna selecionada.



```
SELECT MAX(nome_coluna)
FROM nome_tabela
WHERE condicao;
```

7. Manipulação de dados SQL

7.8 LIKE

O operador **LIKE** é usado em uma cláusula **WHERE** para procurar um padrão especificado em uma coluna.

Existem dois curingas frequentemente usados em conjunto com o operador:

- O sinal de porcentagem (%) representa zero, um ou vários caracteres
- O sinal de sublinhado (_) representa um único caractere

```
SELECT coluna1, coluna2, ...
FROM nome_tabela
WHERE colunaN LIKE padrao;
```

Podemos utilizar como exemplo, um padrão de busca por uma coluna que tenha o valor começando com “Luiza”:

colunaN LIKE ‘Luiza%’



4ªEDIÇÃO

Banco de dados
NoSQL

luiiza
<CODE>

Ementa:

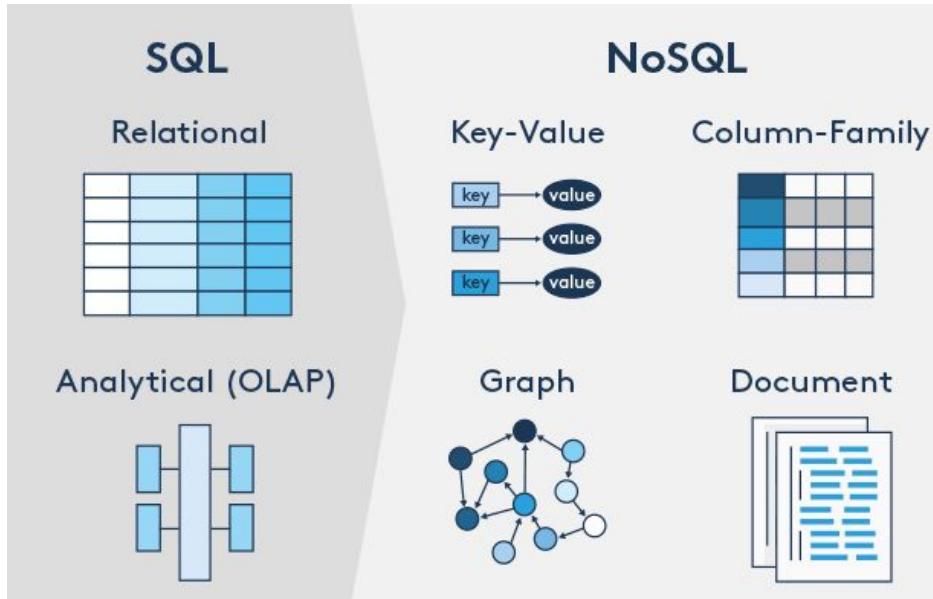
1. Banco de dados não relacional
2. Introdução ao MongoDB
3. Instalando e configurando o MongoDB (VSCode)
4. Base de dados no MongoDB
5. Operações CRUD do MongoDB
6. Operadores de Comparação e Lógicos no MongoDB
7. Agregação no MongoDB
8. Mapeamento SQL para MongoDB

9. Banco de dados não relacional - NoSQL

9. Banco de dados não relacional

9.1 O que é NoSQL?

O termo '**NoSQL**' se refere a tipos **não relacionais de bancos de dados**, e esses bancos de dados armazenam dados em um **formato diferente** das **tabelas relacionais**. No entanto, os bancos de dados NoSQL podem ser consultados usando APIs de linguagem idiomática, linguagens de consulta estruturadas declarativas e linguagens de consulta por exemplo, razão pela qual também são chamados de bancos de dados "não apenas SQL".



9. Banco de dados não relacional

9.2 Para que é usado um banco de dados NoSQL?

Os bancos de dados **NoSQL** são amplamente usados em **aplicativos da web** em tempo real e big data, porque suas principais vantagens são **alta escalabilidade** e **alta disponibilidade**.

Os bancos de dados NoSQL também são a escolha preferida dos desenvolvedores, pois eles **naturalmente aceitam** um **paradigma** de **desenvolvimento ágil**, **adaptando-se** rapidamente aos **requisitos** em **constante mudança**. Os bancos de dados NoSQL permitem que os dados sejam **armazenados** de **maneiras** mais **intuitivas** e **fáceis** de entender, ou mais **próximas** da **maneira** como os **dados são usados** pelos aplicativos - com menos transformações necessárias ao armazenar ou recuperar usando APIs no estilo NoSQL. Além disso, os bancos de dados NoSQL podem aproveitar ao máximo a nuvem para oferecer tempo de inatividade zero.

9. Banco de dados não relacional

9.3 Banco de dados relacional versus banco de dados NoSQL

Os dados em um **RDBMS** são armazenados em objetos de banco de dados que são chamados de **tabelas**. Uma tabela é uma **coleção de entradas de dados relacionadas** e consiste em **colunas** e **linhas**. Esses bancos de dados **requerem** a **definição do esquema** com **antecedência**, ou seja, todas as colunas e seus tipos de dados associados devem ser **conhecidos** de **antemão** para que os **aplicativos** possam **gravar dados** no banco de dados. Eles também armazenam informações **vinculando** várias tabelas por meio do uso de chaves, criando assim um relacionamento entre várias tabelas. No caso mais simples, uma chave é usada para recuperar uma linha específica para que ela possa ser examinada ou modificada.

Por outro lado, em bancos de dados **NoSQL**, os dados podem ser **armazenados sem definir o esquema com antecedência**—o que significa que você tem a **capacidade** de se **mover** e **iterar** rapidamente, **definindo o modelo de dados** à medida que avança. Isso pode ser adequado para requisitos específicos de negócios, seja baseado em gráficos, orientado a colunas, orientado a documentos ou como um armazenamento de valor-chave.

9. Banco de dados não relacional

9.4 Tipos de banco de dados NoSQL

Banco de Documentos: Armazena seus dados em documentos semelhantes aos objetos **JSON (JavaScript Object Notation)**. Possuem normalmente poderosas linguagens de consulta, esses bancos de dados de documentos são ótimos para usos gerais. Eles podem ser facilmente escalados horizontalmente para acomodar grandes volumes de dados. O MongoDB é constantemente classificado como o banco de dados **NoSQL mais popular no mundo**, e é um exemplo de banco de dados de **documentos**. Confira abaixo um exemplo de uma collection (“tabela”) do MongoDB:

```
[{  
    "_id": ObjectId("5e6261a1df9bcf90c29726d4") ,  
    "nome": "Henrique Marques Fernandes" ,  
    "idade": 29  
},  
{  
    "_id": ObjectId("5e6261a1df9bcf90c29726d3") ,  
    "nome": "Terry Crews" ,  
    "idade": 65 ,  
    "pais": "USA"  
}]
```

9. Banco de dados não relacional

9.4 Tipos de banco de dados NoSQL

Chave-Valor: São um tipo mais “simples” de banco de dados, em que cada **item contém chaves e valores**. Esses valores podem ser qualquer tipo de dado, um texto, um número, um JSON e eles podem ser recuperados fazendo referência a sua chave, fazendo com que sua consulta seja muito simples. Esses bancos são ótimos para quando você precisa **armazenar grandes quantidades de dados**, mas **não precisa executar consultas complexas** neles. Os usos mais comuns são para armazenamento de **dados em cache**. **Redis** e **DynamoDB** são provavelmente os bancos mais populares desse tipo.

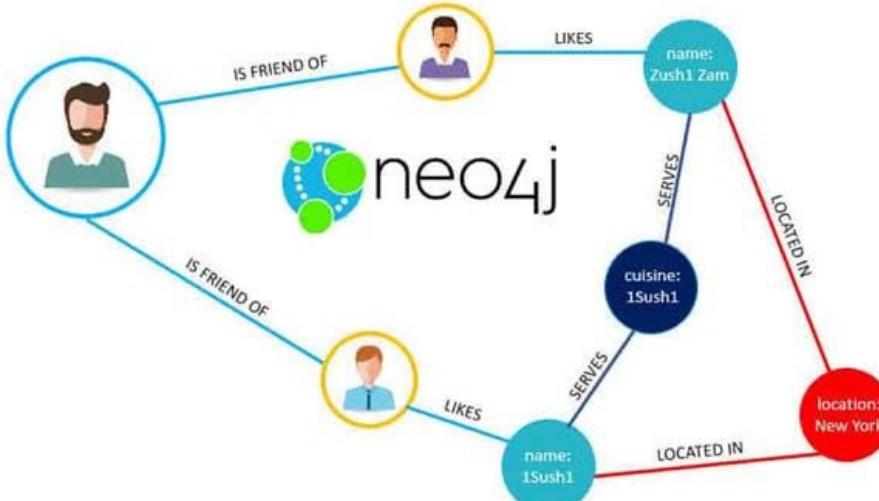
```
1 { "id": 1, "nome": "Terry Crews", "idade": 65, "pais": "USA" }
```

```
2 Henrique Marques Fernandes
```

9. Banco de dados não relacional

9.4 Tipos de banco de dados NoSQL

Grafo: Este banco de dados organiza os dados como **nós** e **relacionamentos**, que mostram as **conexões** entre os **nós**. Isso oferece suporte a uma representação de dados mais rica e completa. Bancos de dados gráficos são aplicados em redes sociais, sistemas de reserva e detecção de fraudes. Exemplo: Neo4j.



9. Banco de dados não relacional

9.4 Tipos de banco de dados NoSQL

Colunar: Ao invés de cada registro da tabela ficar armazenado em uma linha, o registro passa a ser armazenado em colunas separadas. Essa forma de armazenamento tem algumas vantagens, como exemplo a capacidade de compressão dos dados, se formos analisar a compressão de um banco onde os registros são armazenados em linha, encontraremos em uma mesma linha diferentes tipos (domínios) o que torna o processo mais complicado, já no banco orientado a colunas, cada coluna irá conter o mesmo tipo (domínio) de dado.

The diagram illustrates a column-oriented database structure with three columns:

- people_id**: A column where each row contains an **id** and a **value**. The data is as follows:

id	value
0	101
1	102
2	103

- people_name**: A column where each row contains an **id** and a **value**. The data is as follows:

id	value
0	Mary
1	Jhon
2	Paul

- people_age**: A column where each row contains an **id** and a **value**. The data is as follows:

id	value
0	54
1	35
2	22

9. Banco de dados não relacional

9.4 Benefícios de um banco de dados NoSQL

- **Flexibilidade:** Com os bancos de **dados SQL**, os dados são **armazenados** em uma **estrutura predefinida muito mais rígida**. Mas com o **NoSQL**, os dados podem ser **armazenados** de uma forma mais livre, **sem** aqueles **esquemas rígidos**. Este design permite inovação e rápido desenvolvimento de aplicativos. Os desenvolvedores podem se concentrar na criação de sistemas para melhor atender seus clientes, sem preocupar-se com os esquemas. Os bancos de dados NoSQL podem lidar facilmente com qualquer formato de dados, como dados estruturados, semiestruturados e não estruturados em um único armazenamento de dados.
- **Escalabilidade:** Em vez de escalar adicionando mais servidores, os bancos de dados NoSQL podem **escalar usando hardware comum**. Isso tem a capacidade de suportar o aumento do tráfego para atender à demanda com tempo de inatividade zero. Ao expandir, os bancos de dados NoSQL podem se tornar maiores e mais poderosos, e é por isso que eles se tornaram a opção preferida para conjuntos de dados em evolução.

9. Banco de dados não relacional

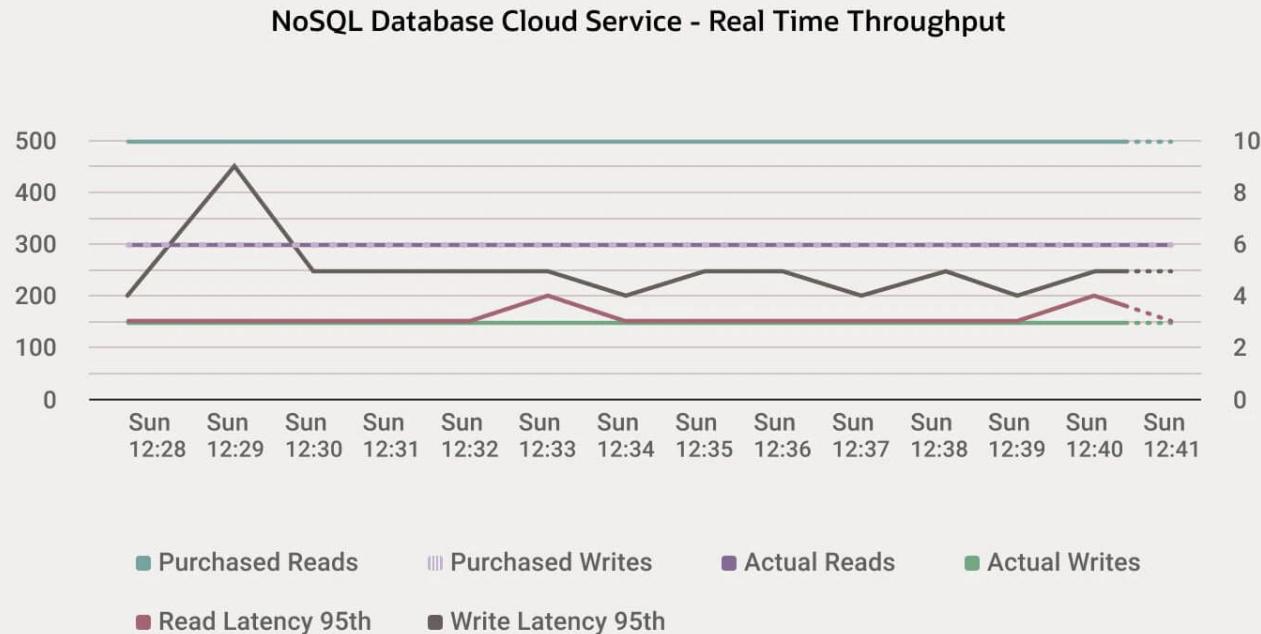
9.4 Benefícios de um banco de dados NoSQL

- **Alto Desempenho:** A arquitetura de expansão de um banco de dados NoSQL pode ser particularmente valiosa quando o volume de dados ou o tráfego aumenta. Essa arquitetura garante **tempos de resposta rápidos** e **previsíveis** em **milissegundos** de um dígito. Os bancos de dados NoSQL também podem ingerir dados e entregá-los de forma rápida e confiável, e é por isso que os bancos de dados NoSQL são usados em aplicativos que coletam terabytes de dados todos os dias, ao mesmo tempo que exigem uma experiência de usuário altamente interativa. No gráfico abaixo, mostramos uma taxa de entrada de 300 leituras por segundo (linha azul) com uma latência de 95 no intervalo de 3-4 ms e uma taxa de entrada de 150 gravações por segundo (linha verde) com uma latência de 95 no intervalo de 4-5ms.

9. Banco de dados não relacional

9.4 Benefícios de um banco de dados NoSQL

- Alto Desempenho



9. Banco de dados não relacional

9.4 Benefícios de um banco de dados NoSQL

- **Disponibilidade:** Os bancos de dados NoSQL **replicam dados automaticamente em vários servidores**, data centers ou recursos de **nuvem**. Por sua vez, isso **minimiza a latência** para os **usuários**, não importa onde eles estejam localizados. Esse recurso também **reduz a carga de gerenciamento do banco de dados**, o que libera tempo para concentrar-se em outras prioridades.
- **Altamente Funcional:** Os bancos de dados NoSQL são projetados para **armazenamentos de dados distribuídos** que têm necessidades de armazenamento de dados **extremamente grandes**. Isso é o que torna o NoSQL a escolha ideal para big data, aplicativos da web em tempo real, cliente 360, compras online, jogos online, Internet of things, redes sociais e aplicativos de publicidade online.

10. Introdução ao MongoDB

10. Introdução ao MongoDB

10.1 - Como funciona o MongoDB

MongoDB é um software de **banco de dados orientado** a documentos livre, de código aberto e multiplataforma, escrito na linguagem C++. Classificado como um programa de **banco de dados NoSQL**, o MongoDB usa **documentos semelhantes a JSON** com **esquemas**. É desenvolvido pela MongoDB Inc. e publicado sob uma combinação da GNU Affero General Public License e Licença Apache.

Os itens não são relacionados, obrigatoriamente, e sua hierarquia é totalmente flexível. Por conta de seu banco de dados NoSQL, as informações são armazenadas nas coleções e documentos. As coleções são subpartes do banco geral, independentes.



mongoDB

10. Introdução ao MongoDB

10.2 - Vantagens do MongoDB

O **MongoDB** contém **coleções**, assim como o banco de dados **PostgreSQL** contém **tabelas**. A sua vantagem é a **permissão** para **criar vários bancos de dados e várias coleções dentro do principal**.

Na **coleção**, estão **documentos** que contêm os **dados** que vamos **armazenar** no **banco** do MongoDB, e uma **única coleção** pode **conter vários documentos**. Não existe esquema de tipo, isso significa que **não é necessário** que **um documento seja semelhante ao outro**.



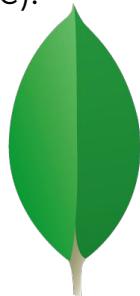
mongoDB

10. Introdução ao MongoDB

10.2 - Vantagens do MongoDB

Nos **documentos**, pode-se armazenar **dados aninhados**. Essa conexão de dados permite **criar relações complexas** entre eles e **armazená-los** no mesmo **documento**, o que torna o trabalho e a **busca mais eficientes** em **comparação** com o **SQL**. Além disso, podemos citar:

- **Não precisa projetar o esquema** do banco de dados ao trabalhar com o mongoDB;
- **Fornece** grande **flexibilidade** para os **campos** nos **documentos**;
- Trabalha com **dados heterogêneos**;
- Não requer **nenhuma** adição ou **injeção** de **SQL**;
- Facilmente **integrável** com o **Big Data Hadoop** (com diversas versões open source).



mongoDB

10. Introdução ao MongoDB

10.3 - Desvantagens do MongoDB

- Utiliza **muita memória** para **armazenar e estocar dados**;
- **Límite de 16 MB** de dados para armazenar nos documentos;
- **Límite para aninhar dados nos arquivos BSON** em até **100 níveis**.



mongoDB

11. Instalando e configurando o MongoDB

11. Instalando o MongoDB

11.1 Instalando o MongoDB no VSCode

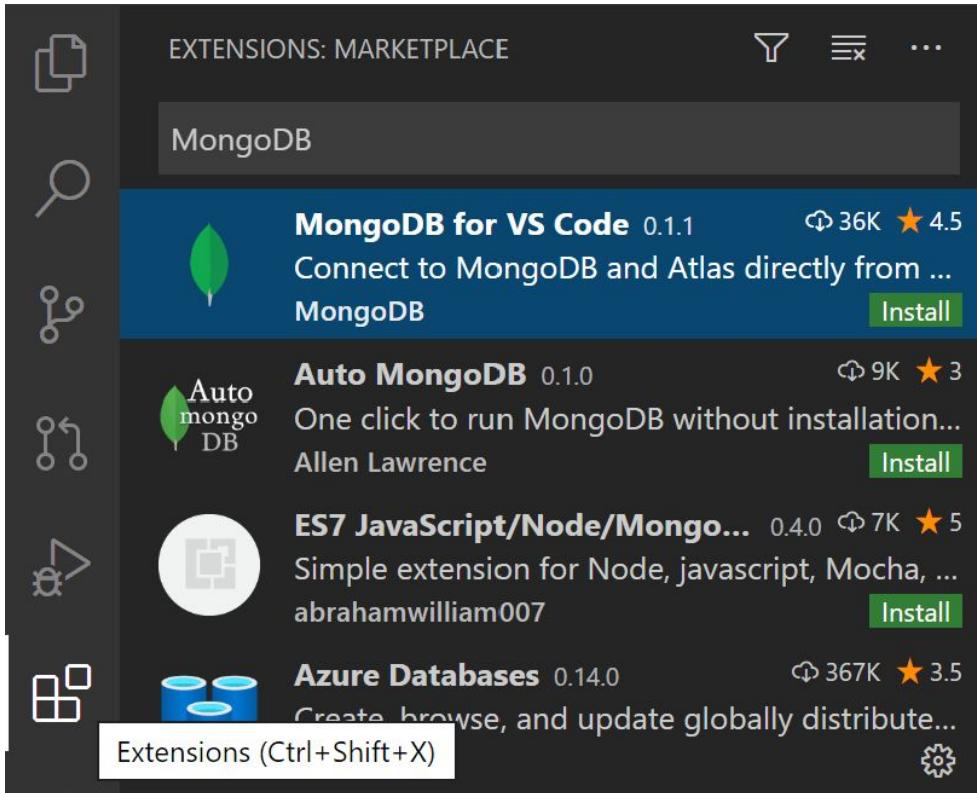
O Visual Studio Code tem um ótimo suporte para trabalhar com bancos de dados MongoDB , seja sua própria instância ou no Azure com MongoDB Atlas . Com a extensão MongoDB para VS Code , você pode criar, gerenciar e consultar bancos de dados MongoDB de dentro do VS Code.

Instale a extensão

O suporte do MongoDB para VS Code é fornecido pela extensão MongoDB para VS Code . Para instalar a extensão MongoDB para VS Code, abra a visualização Extensions pressionando Ctrl+Shift+X e procure por 'MongoDB' para filtrar os resultados. Selecione a extensão MongoDB para VS Code .

11. Instalando o MongoDB

11.1 Instalando o MongoDB no VSCode

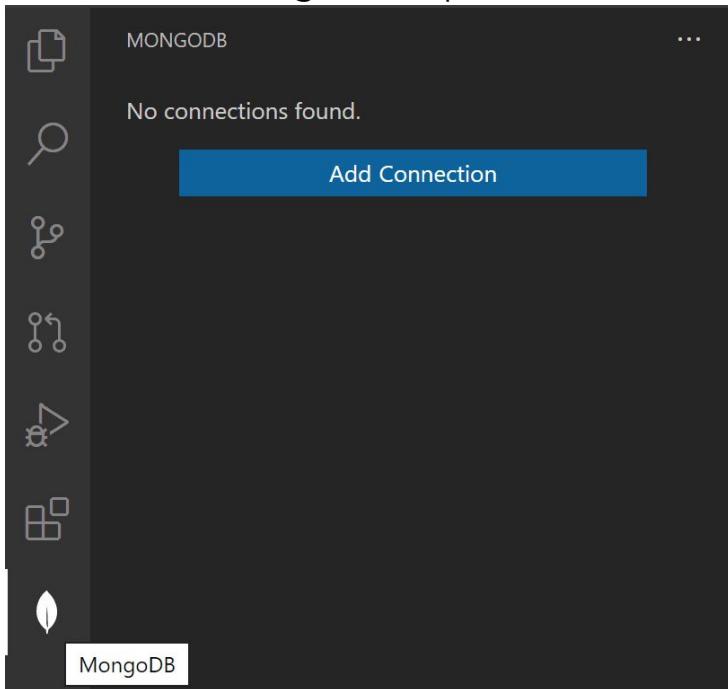


11. Instalando o MongoDB

11.1 Instalando o MongoDB no VSCode

Conecte-se ao MongoDB

Depois de instalar a extensão MongoDB para VS Code, você notará que há uma nova visualização da barra de atividades do MongoDB . Selecione a visualização do MongoDB e você verá o MongoDB Explorer.



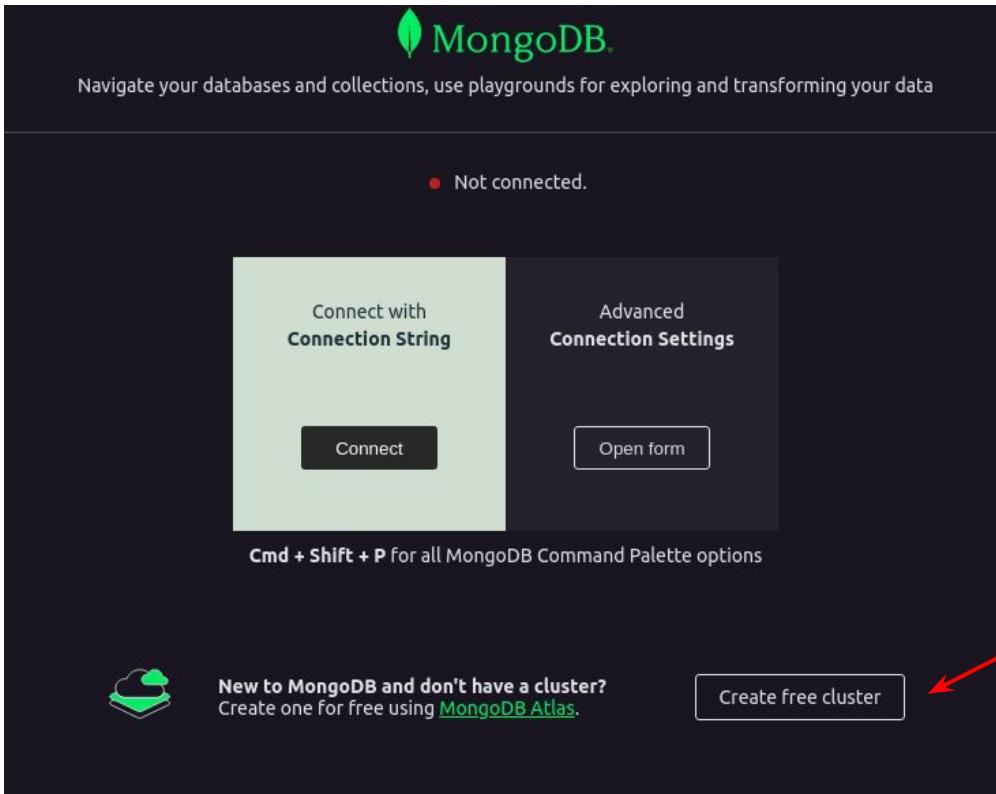
11. Configurando o MongoDB

11.1 Configurando o MongoDB no VSCode

Para se conectar a um banco de dados MongoDB, selecione Adicionar conexão e insira os detalhes da conexão para o banco de dados e, em seguida , Conectar , o padrão é um servidor MongoDB local em `mongodb://127.0.0.1:27017`. Você também pode inserir uma string de conexão, clicar no link "conectar com uma string de conexão" e colar a string de conexão.

11. Configurando o MongoDB

11.2 Cluster atlas MongoDB no VSCode



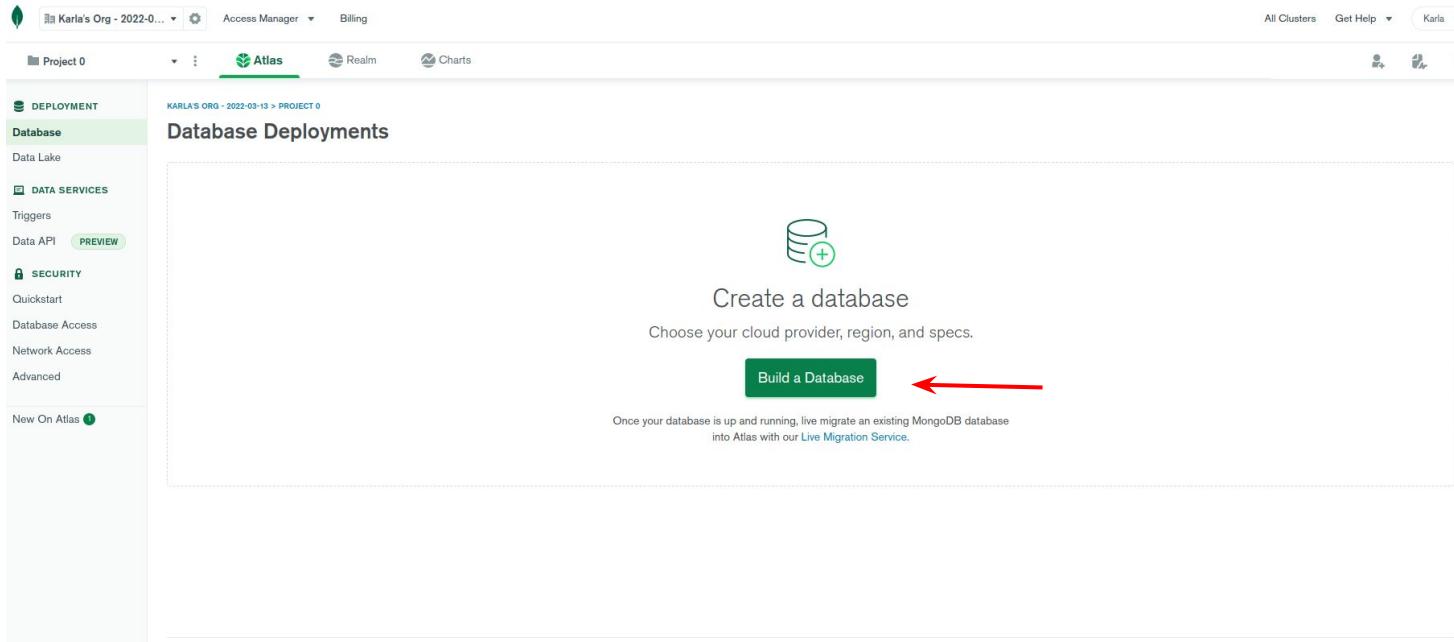
Passo 1

11. Configurando o MongoDB

11.2 Cluster atlas MongoDB no VSCode

Passo 2: Crie uma conta e faça login.

Passo 3: Clique em Build a Database.



The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with sections like Project 0, Deployment (Database selected), Data Lake, Data Services (Triggers, Data API PREVIEW), Security (Quickstart, Database Access, Network Access, Advanced), and New On Atlas. The main area is titled "Database Deployments" and "KARLAS ORG - 2022-03-13 > PROJECT 0". It features a "Create a database" section with a green icon of two overlapping circles with a plus sign. Below it, the text says "Choose your cloud provider, region, and specs." and a green "Build a Database" button. A red arrow points to this button. At the bottom, there's a note: "Once your database is up and running, live migrate an existing MongoDB database into Atlas with our Live Migration Service."

11. Configurando o MongoDB

11.2 Cluster atlas MongoDB no VSCode

The screenshot shows the MongoDB Atlas deployment options page. It features three main deployment options:

- Serverless**: For serverless applications. Starting at \$0.30/1M reads.
- Dedicated**: For production applications. Starting at \$0.08/hr*, estimated cost \$56.94/month.
- Shared**: For learning and exploring MongoDB in a cloud environment. Basic configuration options. Starting at FREE.

A red arrow points to the "Create" button for the Shared cluster.

MongoDB.
MONGODB ATLAS

Deploy a cloud database

Experience the best of MongoDB on AWS, Azure, and Google Cloud. Choose a deployment option to get started.

PREVIEW

Serverless

For serverless applications that aren't critical with variable traffic. Minimal configuration required.

- ✓ Pay only for the operations you run
- ✓ Resources scale seamlessly to meet your workload
- ✓ Always-on security and backups

Create

Starting at
\$0.30/1M reads

ADVANCED

Dedicated

For production applications with sophisticated workload requirements. Advanced configuration controls.

- ✓ Network isolation and fine-grained access controls
- ✓ On-demand performance advice
- ✓ Multi-region and multi-cloud options available

Create

Starting at
\$0.08/hr*
*estimated cost \$56.94/month

FREE

Shared

For learning and exploring MongoDB in a cloud environment. Basic configuration options.

- ✓ No credit card required to start
- ✓ Explore with sample datasets
- ✓ Upgrade to dedicated clusters for full functionality

Create

Starting at
FREE

11. Configurando o MongoDB

11.2 Cluster atlas MongoDB no VSCode

The screenshot shows the MongoDB Atlas cluster creation interface. At the top, there are three tabs: 'PREVIEW' (Serverless), 'Dedicated', and 'FREE Shared'. A red arrow points to the 'Shared' tab. Below the tabs, a message states: 'For learning and exploring MongoDB in a sandbox environment. Basic configuration controls. No credit card required to start. Upgrade to dedicated clusters for full functionality. Explore with sample datasets. Limit of one free cluster per project.' Under 'Cloud Provider & Region', 'AWS, Sao Paulo (sa-east-1)' is selected. A red arrow points to the 'aws' icon. Below this, there are tabs for 'NORTH AMERICA', 'EUROPE', 'AUSTRALIA', 'ASIA', and 'SOUTH AMERICA'. Under 'NORTH AMERICA', 'Sao Paulo (sa-east-1)' is highlighted with a green border and a red arrow pointing to it. Other regions listed include N. Virginia (us-east-1), Oregon (us-west-2), Ohio (us-east-2), N. California (us-west-1), Montreal (ca-central-1), and Sao Paulo (sa-east-1). Under 'EUROPE', regions include Paris (eu-west-3), Frankfurt (eu-central-1), Stockholm (eu-north-1), London (eu-west-2), Milan (eu-south-1), and Bahrain (me-south-1). Under 'ASIA', regions include Sydney (ap-southeast-2), Hong Kong (ap-east-1), Seoul (ap-northeast-2), Singapore (ap-southeast-1), Tokyo (ap-northeast-1), and Mumbai (ap-south-1). Under 'SOUTH AMERICA', regions include Sao Paulo (sa-east-1).

11. Configurando o MongoDB

11.2 Cluster atlas MongoDB no VSCode

The screenshot shows the MongoDB Atlas Security Quickstart interface. On the left sidebar, under the SECURITY section, 'Quickstart' is selected. The main content area is titled 'Security Quickstart' and contains the following text: 'To access data stored in Atlas, you'll need to create users and set up network security controls. Learn more about security setup'. Below this, a question is displayed: '1 How would you like to authenticate your connection?'. A note states: 'Your first user will have permission to read and write any data in your project.' Two authentication methods are shown: 'Username and Password' (selected) and 'Certificate'. A red arrow points from the text 'Certificate' to its corresponding button. The 'Username and Password' section contains fields for 'Username' (karlapereira) and 'Password' (*****). A red arrow points from the 'karlapereira' field to the 'Username' label. Another red arrow points from the '*****' field to the 'Secure Password' link. A 'Create User' button at the bottom is also highlighted with a red arrow.

11. Configurando o MongoDB

11.2 Cluster atlas MongoDB no VSCode

The screenshot shows the MongoDB Atlas dashboard for the project 'KARLA'S ORG - 2022-03-13 > PROJECT 0'. The left sidebar has tabs for Project 0, DEPLOYMENT (selected), DATA LAKE, DATA SERVICES, SECURITY, and New On Atlas. The main area shows 'Database Deployments' for 'Cluster0'. A red arrow points to the 'Database' tab in the sidebar. Another red arrow points to the 'Connect' button in the Cluster0 card. The Cluster0 card displays metrics: R 0, W 0, Connections 0, In 0.0 B/s, Out 0.0 B/s, Data Size 0.0 B / 512.0 MB (0%), and a note about enhancing experience with an Upgrade button. Below the card is a table with columns: VERSION, REGION, CLUSTER TIER, TYPE, BACKUPS, LINKED REALM APP, and ATLAS SEARCH. The table data is: 5.0.6, AWS / Sao Paulo (sa-east-1), M0 Sandbox (General), Replica Set - 3 nodes, Inactive, None Linked, Create Index.

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED REALM APP	ATLAS SEARCH
5.0.6	AWS / Sao Paulo (sa-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Create Index

11. Configurando o MongoDB

11.2 Cluster atlas MongoDB no VSCode

Connect to Cluster0

Setup connection security > Choose a connection method > Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more ↗](#)

You can't connect yet. Set up your firewall access below.

1 Add a connection IP address

Add Your Current IP Address Add a Different IP Address Allow Access from Anywhere

2 Create a Database User

✓ A MongoDB user has been added to this project. Not yours? Create one in the [MongoDB Users tab](#).

You'll need your MongoDB user's credentials in the next step.

[Close](#) Choose a connection method



11. Configurando o MongoDB

11.2 Cluster atlas MongoDB no VSCode

Connect to Cluster0

✖ Setup connection security Choose a connection method Connect

You can't connect yet. Set up your firewall access in the first step.

Choose a connection method [View documentation](#)

Get your pre-formatted connection string by selecting your tool below.

-  **Connect with the MongoDB Shell**
Interact with your cluster using MongoDB's interactive Javascript interface >
-  **Connect your application**
Connect your application to your cluster using MongoDB's native drivers >
-  **Connect using MongoDB Compass**
Explore, modify, and visualize your data with MongoDB's GUI >

Go Back Close



11. Configurando o MongoDB

11.2 Cluster atlas MongoDB no VSCode

Connect to Cluster0

✖ Setup connection security ✓ Choose a connection method Connect

You can't connect yet. Set up your firewall access in the first step.

I do not have MongoDB Compass I have MongoDB Compass

① Choose your version of Compass:
1.12 or later ▾
See your Compass version in "About Compass"

② Copy the connection string, then open MongoDB Compass.
mongodb+srv://karlapereira:<password>@cluster0.kjg0i.mongodb.net/test  

You will be prompted for the password for the **karlapereira** user's (Database User) username.
When entering your password, make sure that any special characters are **URL encoded**.

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back Close

11. Configurando o MongoDB

11.2 Cluster atlas MongoDB no VSCode

We are deploying your changes (current action: configuring MongoDB)

KARLA'S ORG - 2022-03-13 > PROJECT 0

Network Access

IP Access List Peering Private Endpoint

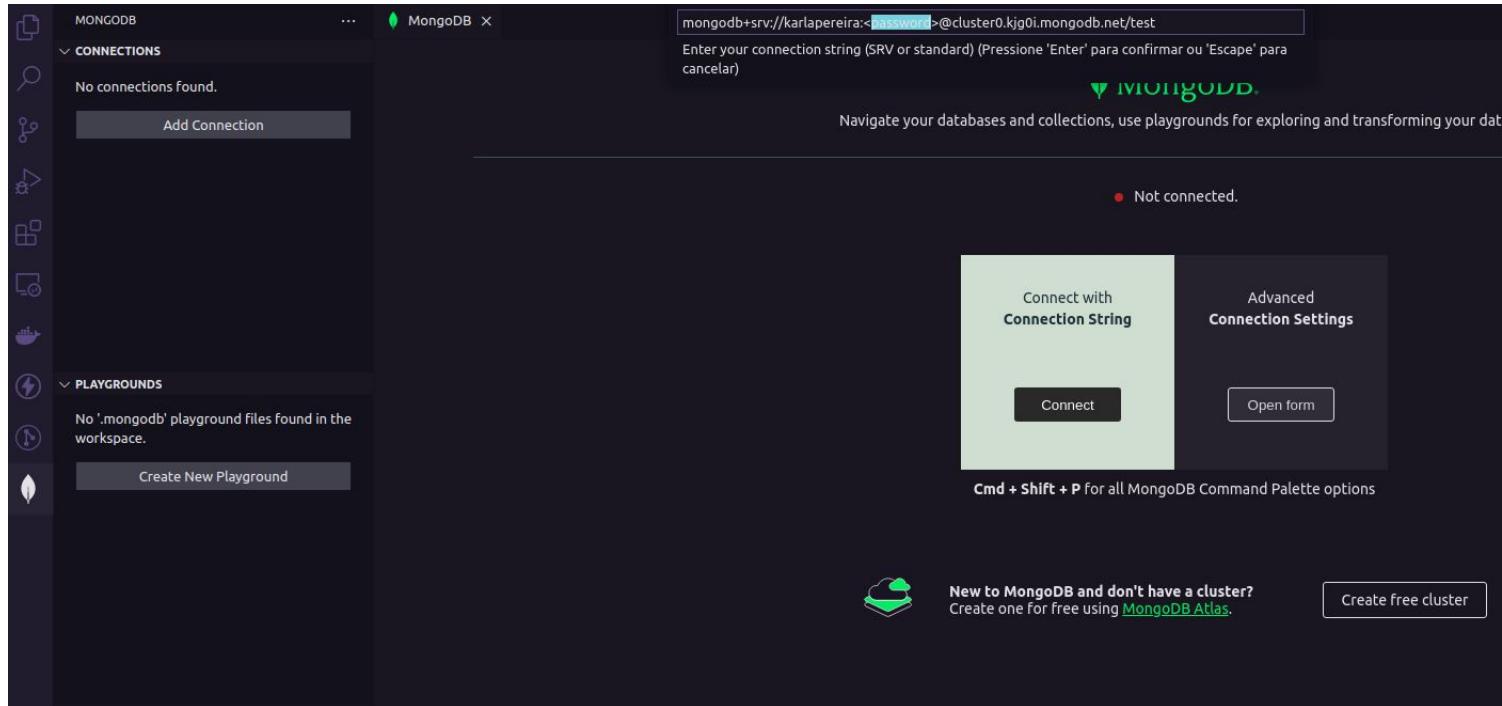
You will only be able to connect to your cluster from the following list of IP Addresses:

IP Address	Comment	Status	Actions
186.235.3.82/32 (includes your current IP address)	Karla-Notebook	Pending	<button>EDIT</button> <button>DELETE</button>

11. Configurando o MongoDB

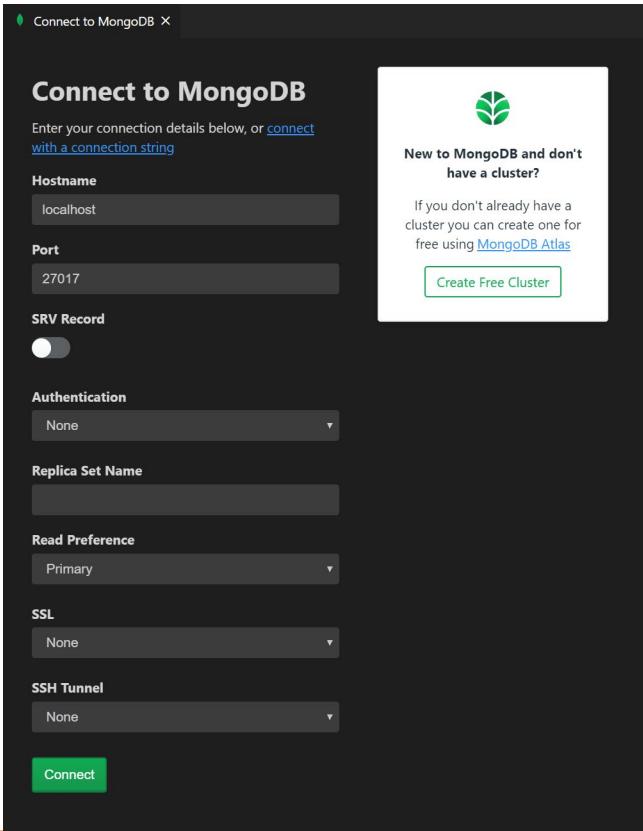
11.2 Cluster atlas MongoDB no VSCode

Após copiar a Connection String, trocar a parte do <password> pela **senha criada** no atlas e conectar no vs code



11. Configurando o MongoDB

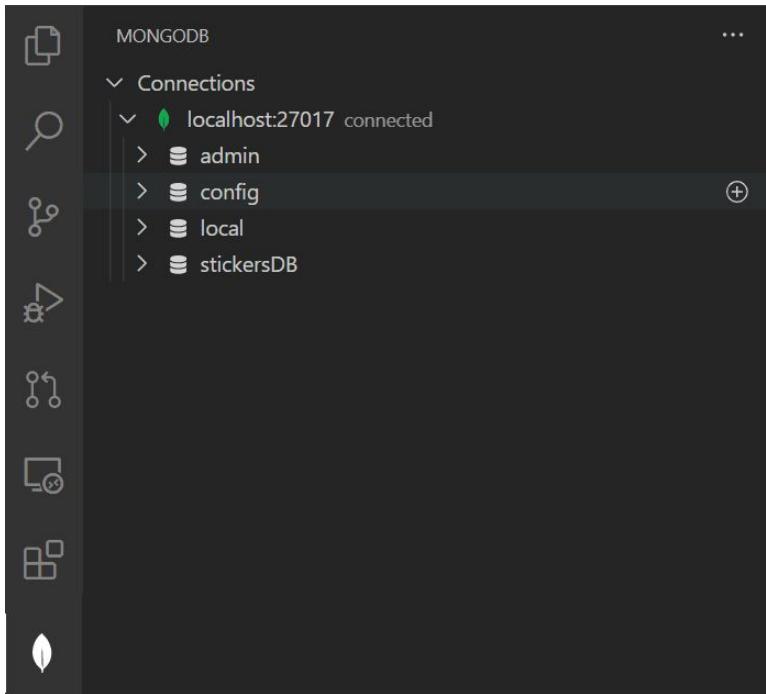
11.3 MongoDB Local no VSCode (Opcional)



11. Configurando o MongoDB

11.4 Usando o MongoDB VS Code

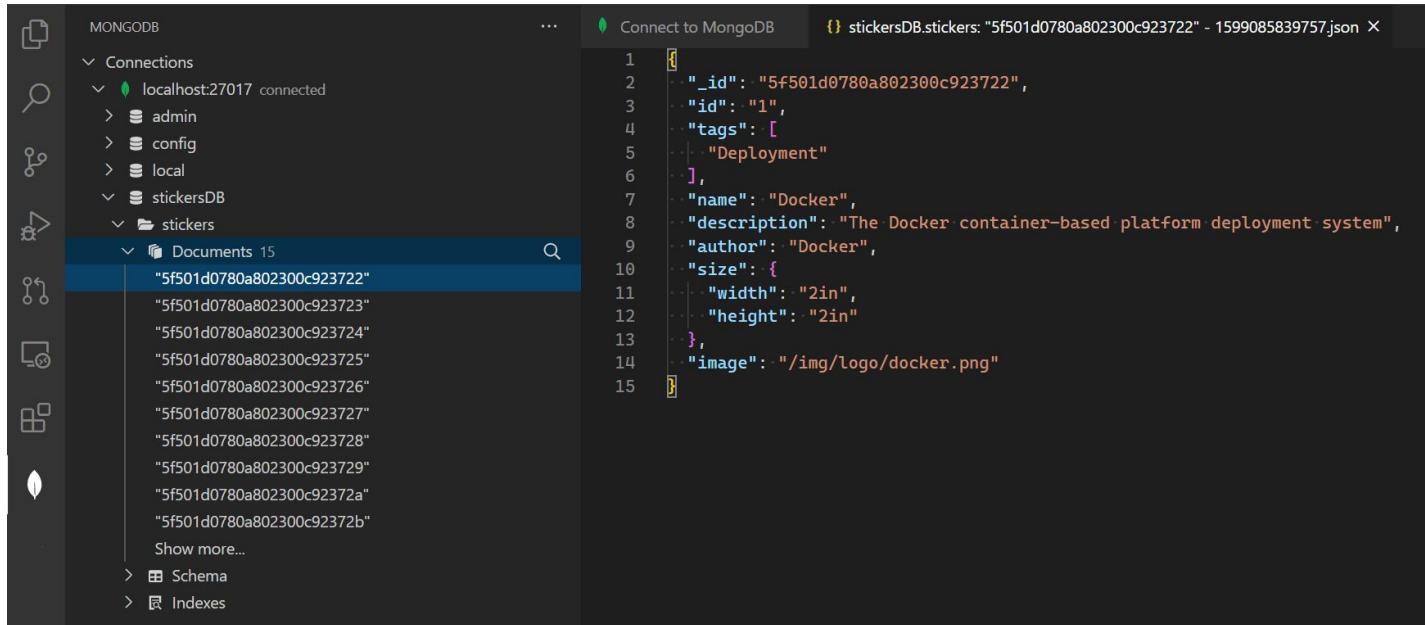
Uma vez anexado, você pode trabalhar com o servidor MongoDB, gerenciando bancos de dados, coleções e documentos do MongoDB.



11. Configurando o MongoDB

11.4 Usando o MongoDB VS Code

Você pode expandir bancos de dados para visualizar suas coleções com seus esquemas e índices e pode selecionar Documentos MongoDB individuais para visualizar seu JSON.



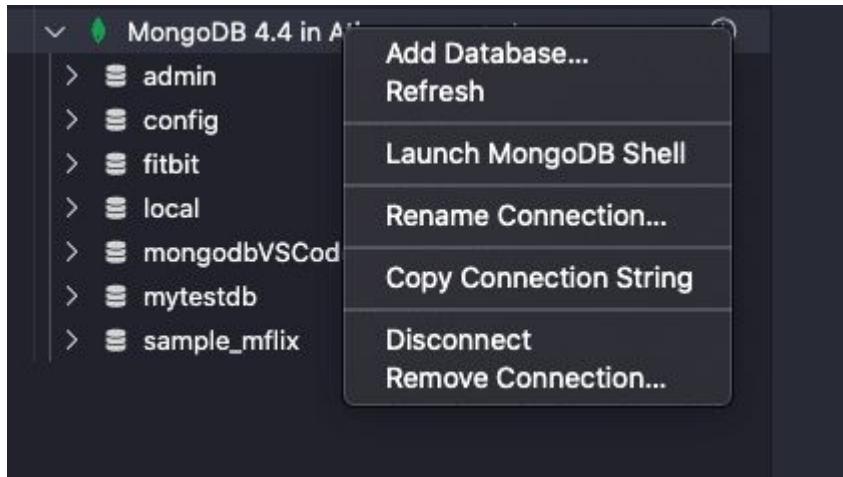
The screenshot shows the MongoDB extension interface in VS Code. On the left, there's a sidebar with icons for file operations like copy, search, and schema/indexes. The main area shows a tree view of database connections and collections. Under 'Connections', 'localhost:27017' is connected, showing collections 'admin', 'config', 'local', and 'stickersDB'. 'stickersDB' contains a 'stickers' folder with 15 documents. One document is selected, showing its full JSON content in the center-right panel. The JSON is as follows:

```
1  [
2    {
3      "_id": "5f501d0780a802300c923722",
4      "id": "1",
5      "tags": [
6        "Deployment"
7      ],
8      "name": "Docker",
9      "description": "The Docker container-based platform deployment system",
10     "author": "Docker",
11     "size": {
12       "width": "2in",
13       "height": "2in"
14     },
15     "image": "/img/logo/docker.png"
}
```

11. Configurando o MongoDB

11.4 Usando o MongoDB VS Code

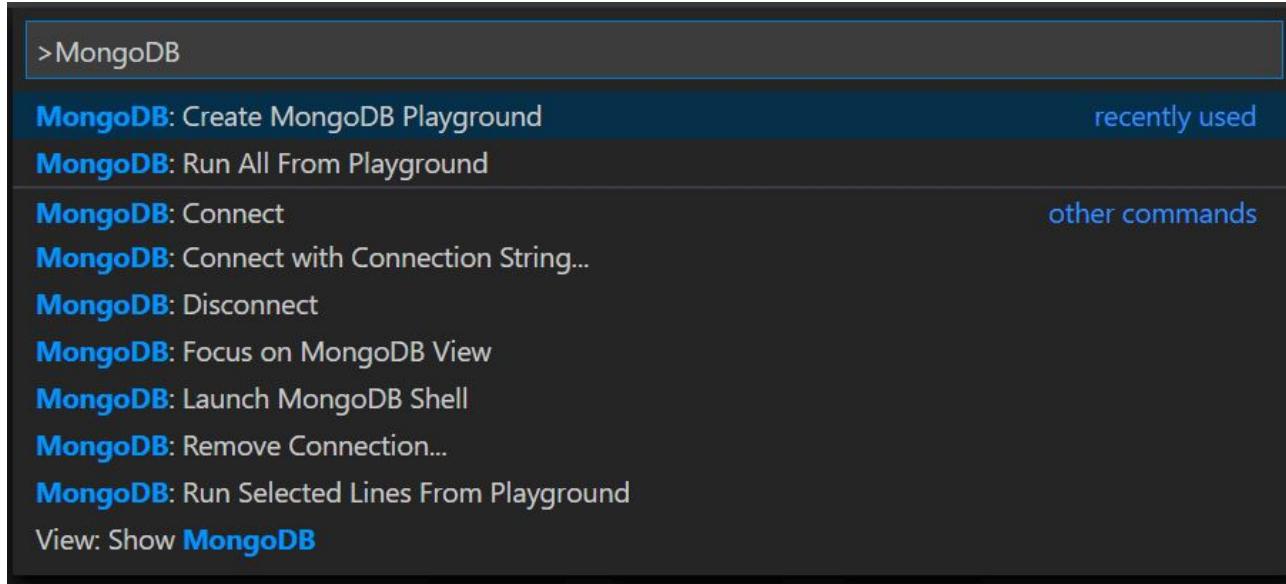
Você também pode anexar um shell do MongoDB à conexão ativa, simplesmente clicando com o botão direito do mouse na própria conexão.



11. Configurando o MongoDB

11.4 Usando o MongoDB VS Code

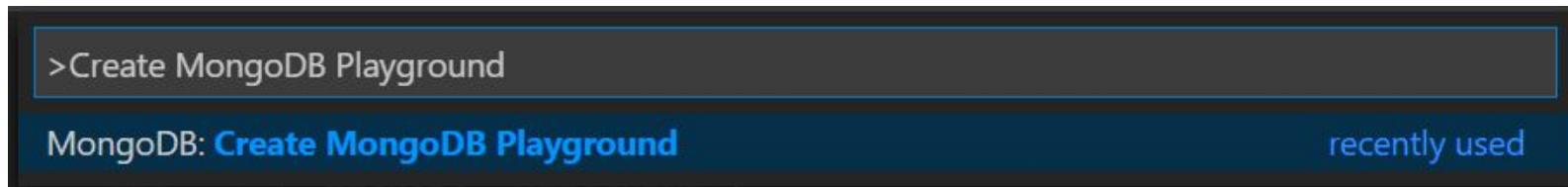
Comandos do MongoDB: Existem comandos específicos do MongoDB disponíveis na Paleta de Comandos do VS Code (Ctrl+Shift+P), bem como nos menus de contexto do Explorer.



11. Configurando o MongoDB

11.4 Usando o MongoDB VS Code

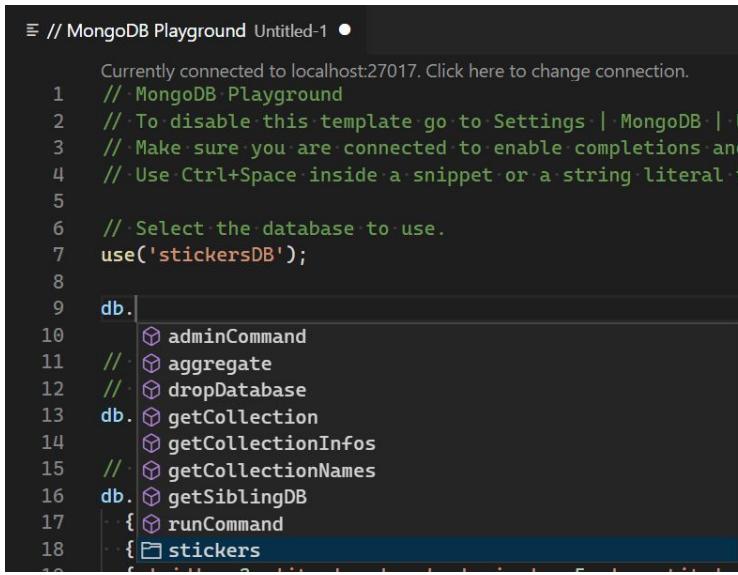
Um dos recursos mais poderosos da integração do VS Code MongoDB é o Mongo Playgrounds . O **Playgrounds** permite **criar**, **executar** e **salvar** **comandos do MongoDB** a partir de um editor do VS Code. Crie um novo playground com o comando MongoDB: Create MongoDB Playground .



11. Configurando o MongoDB

11.4 Usando o MongoDB VS Code

Em um playground, você pode fazer referência a entidades e comandos do MongoDB e obter o IntelliSense avançado à medida que digita. Playgrounds são úteis para prototipagem de operações e consultas de banco de dados. Execute as linhas selecionadas nas consultas do playground com o comando MongoDB: Run Selected Lines From Playground .



The screenshot shows a code editor window titled "MongoDB Playground Untitled-1". It displays the following MongoDB shell script:

```
// MongoDB Playground Untitled-1 ●  
// Currently connected to localhost:27017. Click here to change connection.  
1 // MongoDB Playground  
2 // To disable this template go to Settings | MongoDB | Use Template  
3 // Make sure you are connected to enable completions and suggestions.  
4 // Use Ctrl+Space inside a snippet or a string literal to trigger completions.  
5  
6 // Select the database to use.  
7 use('stickersDB');  
8  
9 db.  
10 db.adminCommand  
11 db.aggregate  
12 db.dropDatabase  
13 db.getCollection  
14 db.getCollectionInfos  
15 db.getCollectionNames  
16 db.getSiblingDB  
17 { runCommand  
18 { stickers
```

The cursor is positioned at the end of the line "db.". A dropdown menu is open, showing suggestions for MongoDB methods: "adminCommand", "aggregate", "dropDatabase", "getCollection", "getCollectionInfos", "getCollectionNames", "getSiblingDB", "runCommand", and "stickers". The "runCommand" suggestion is highlighted.

12. Base de dados no MongoDB

12. Base de dados no MongoDB

12.1 Esquema Flexível NoSQL

Diferentemente dos bancos de dados SQL, onde você deve determinar e declarar o esquema de uma tabela antes de inserir os dados, as coleções do **MongoDB**, por padrão, não exigem que seus documentos tenham o mesmo esquema. Isso é:

- Os documentos em uma única coleção **não precisam** ter o **mesmo conjunto de campos** e o **tipo** de **dados** de um campo pode diferir entre os documentos de uma coleção.
- Para **alterar** a **estrutura** dos **documentos** em uma **coleção**, como **adicionar** novos campos, **remover** campos existentes ou **alterar** os valores dos campos para um novo tipo, **atualize os documentos para a nova estrutura**.

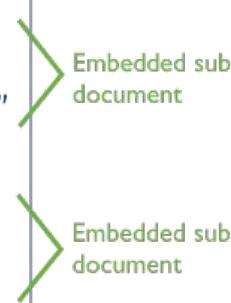
12. Base de dados no MongoDB

12.2 Estrutura do documento

Dados aninhados:

Documentos incorporados **capturam relacionamentos** entre dados **armazenando dados relacionados** em uma **única estrutura de documento**. Os documentos do MongoDB possibilitam incorporar estruturas de documentos em um campo ou array dentro de um documento. Esses modelos de dados **desnormalizados** permitem que os aplicativos recuperem e manipulem dados relacionados em uma única operação de banco de dados.

```
{  
  _id: <ObjectId1>,  
  username: "123xyz",  
  contact: {  
    phone: "123-456-7890",  
    email: "xyz@example.com"  
  },  
  access: {  
    level: 5,  
    group: "dev"  
  }  
}
```



Embedded sub-document

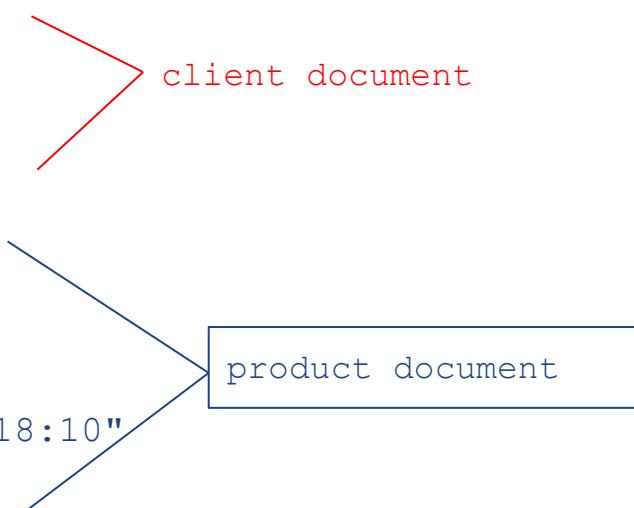
Embedded sub-document

12. Base de dados no MongoDB

12.2 Estrutura do documento

Dados aninhados:

```
wishlist document
{
  _id: 325123,
  name: "Favorital",
  client: {
    name: "Luiza",
    email: "luiza@magalu.com.br"
  },
  products = [
    {
      name: "Notebook",
      price: 5780.00,
      quantity: 18,
      updated_at: "2022/03/15 18:10"
    }
  ]
}
```

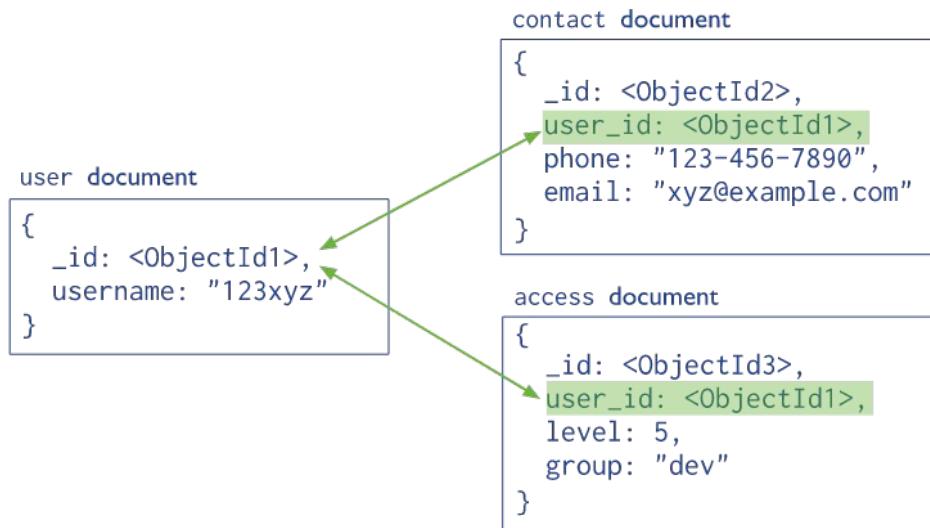


12. Base de dados no MongoDB

12.2 Estrutura do documento

Referências:

As referências **armazenam** os **relacionamentos entre os dados** incluindo **links** ou **referências** de um **documento** para outro. Os aplicativos podem resolver essas referências para acessar os dados relacionados. Em geral, esses são modelos de dados **normalizados**.



12. Base de dados no MongoDB

12.2 Estrutura do documento

Referências:

```
// client document
{
  _id: 986,
  name: "Luiza",
  email: "luiza@magalu.com.br"
}
```

```
// wishlist document
{
  _id: 325123,
  name: "Favorital",
  client_id: 986,
  id_products = [1]
}
```

```
// product document
{
  _id: 1,
  name: "Notebook",
  price: 5780.00,
  quantity: 18,
  updated_at: "2022/03/15 18:10"
}
```

12. Base de dados no MongoDB

12.3 Características do ObjectId

O campo `_id` é um dado do tipo **ObjectId**, e ele tem algumas funções padrões primárias:

- `_id` é a **chave primária** de todos os elementos de uma coleção, isso permite que os registros sejam diferenciados por padrão.
- `_id` é um campo indexado automaticamente. As pesquisas que especificam `{_id: }` referem-se ao índice `_id` como guia.
- Em termos de arquitetura, por padrão, o campo `_id` é um `ObjectID`, um dos tipos `BSON` do `MongoDB`. Os usuários também podem substituir `_id` por algo diferente de um `ObjectID`, embora não seja muito recomendado.

12. Base de dados no MongoDB

12.4 Estruturas do ObjectId

Um ObjectId é um tipo BSON binário de 12 bytes representados em 24 caracteres hexadecimais:

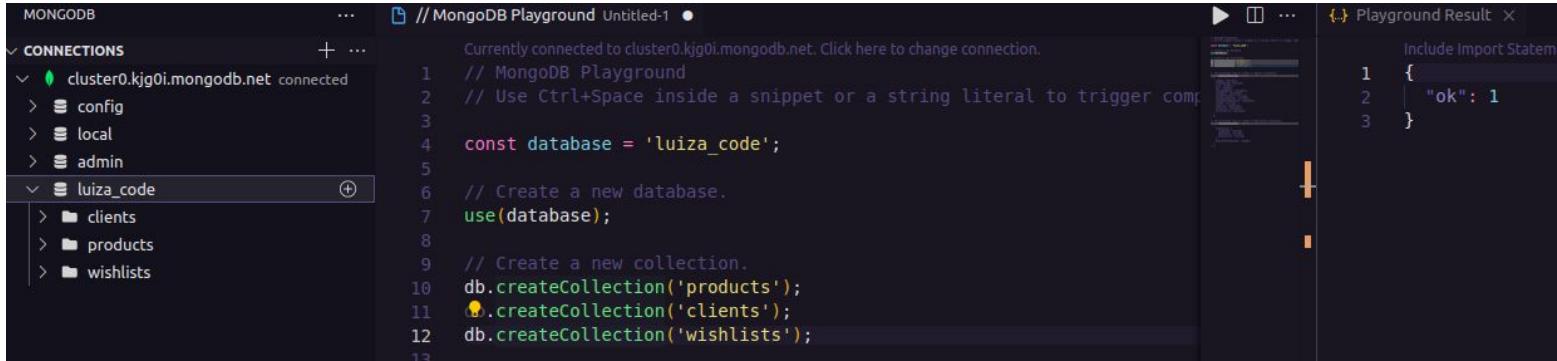
```
{  
    "_id": ObjectId("54759eb3c090d83494e2d804")  
}
```

Esses bytes são gerados automaticamente e separados em grupos com funcionalidades específicas:

Tamanho	Descrição
4 bytes	4 bytes que representa os segundos desde a época do Unix
3 bytes	3-byte com identificador de máquina
2 bytes	2-byte com o identificador único do processo
3 bytes	3-byte contador que começa com um número aleatório por coleção

12. Base de dados no MongoDB

12.3 Criando a base de dados e as coleções



The screenshot shows the MongoDB Playground interface. On the left, the 'CONNECTIONS' sidebar lists 'cluster0.kjg0i.mongodb.net connected' and a local database named 'luiza_code'. Inside 'luiza_code', there are three collections: 'clients', 'products', and 'wishlist'. The main workspace contains a code editor with the following MongoDB shell script:

```
// MongoDB Playground
// Use Ctrl+Space inside a snippet or a string literal to trigger completion
const database = 'luiza_code';
use(database);
db.createCollection('products');
db.createCollection('clients');
db.createCollection('wishlist');
```

The 'Playground Result' panel on the right shows the response from the MongoDB server:

```
{ "ok": 1 }
```

```
use("database"); //cria e/ou altera para o banco
```

```
db.createCollection("coleção"); //cria uma nova coleção
```

```
db.nomedacollection.drop() //deleta uma coleção
```

13. Operações CRUD do MongoDB

13. Operações CRUD do MongoDB

13.1 CRUD

As operações CRUD criam , leem , atualizam e excluem **documentos**. No MongoDB não usamos querys igual ao SQL, usamos métodos das classes do banco ou coleções.

13.2 Operações de criação

As operações de criação ou inserção adicionam novos documentos a uma coleção . Se a coleção não existir no momento, as operações de inserção criarão a coleção.

O MongoDB fornece os seguintes métodos para inserir documentos em uma coleção:

```
db.collection.insertOne({nome: "Luiza", idade: 30, estudante: false})  
db.collection.insertMany([  
    {},  
    {}  
])
```

13. Operações CRUD do MongoDB

13.2 Operações de criação

```
db.users.insertOne(  
  {  
    name: "sue",  
    age: 26,  
    status: "pending"  
  }  
)
```

← collection

← field: value } document

← field: value

← field: value

13. Operações CRUD do MongoDB

13.3 Operações de leitura

As operações de leitura **recuperam documentos** de uma **coleção**; ou seja, consultar uma coleção de documentos. O MongoDB fornece os seguintes métodos para ler documentos de uma coleção:

db.collection.**find()**

```
>> db.clients.find()
{ '_id': 3, 'name': 'Luiza', 'email': 'luiza@gmail.com' }
```

13. Operações CRUD do MongoDB

13.3 Operações de leitura

Para indentá-lo facilitando uma melhor leitura podemos usar o método pretty.

```
>> db.clients.find().pretty()
{
  '_id': 3,
  'name': 'Luiza',
  'email': 'luiza@gmail.com'
}
```

O campo **_id**, esse é o **identificador** de nosso **documento**, quando não o passamos por parâmetro no find, o próprio Mongo o cria.

O método **findOne** que retorna somente 1 documento, aquele com a característica mais próxima a busca referenciada. Sua sintaxe é a mesma do método find.

13. Operações CRUD do MongoDB



13.3 Operações de leitura

Projection

O projection é um objeto que passamos como parâmetro do find, com o intuito de obter somente do document os campos que precisamos, isso é muito útil quando temos documentos com inúmeros campos e precisamos tratar somente alguns dados, isso é ótimo para a performance.

```
db.usuarios.find(  
  {_id:3},  
  {_id:0, usuario:1, senha:1 }  
)
```

O projection retorna somente os campos em que setamos como 1 ou true, não é preciso colocar todos os campos do document no objeto como true ou false, se você não irá precisar do campo _id então você deve setá-lo como 0 ou false, pois por padrão ele é tido como true, é importante a remoção do campo _id quando não for usá-lo até mesmo por uma questão de segurança.

13. Operações CRUD do MongoDB

13.3 Operações de leitura

operadores gt e lt

Existem alguns tipos de chaves que usamos como uma espécie de filtros para o find, como por exemplo:

gt (greater than) maior que

gte (greater than or equals) maior ou igual a

lt (less than) menor que

lte (less than or equals) menor ou igual a

```
db.alunos.find(  
    { notas.matematica: {$gte: 70} }  
)
```

13. Operações CRUD do MongoDB

13.3 Operações de leitura

regex

Você também pode usar **expressões regulares** nos comandos do Mongo Shell.

```
>> db.alunos.find({ nome: {$regex: "e"} })
// retorna todos os alunos que possuem a letra "e" em algum lugar
do nome
```

```
>> db.alunos.find({ nome: {$regex: "^A"} })
// retorna todos os alunos que começam com a letra "A" maiúsculo
```

```
>> db.alunos.find({ nome: {$regex: "e\$"} })
// retorna todos os alunos que termina com a letra "e"
```

13. Operações CRUD do MongoDB

13.3 Operações de leitura

operadores lógicos e / ou

Os operadores lógicos são representados pela sintaxe \$and / \$or e requerem um array, dentro desse array são passadas todas as condições para serem analisadas. O operador \$and retorna o documento buscado, somente se todas as queries passadas forem verdadeiras. E o operador \$or retorna o documento se pelo menos uma query for verdadeira.

Ex: db.pessoas.**find**({ \$or: [**condição 1, condição 2**] })

```
>> db.alunos.find(  
  { $or: [ { nome: {$gt: "D"}, email: {$exists: true} } ] }  
)
```

// retorna todos os alunos que o nome começa a partir da letra D do alfabeto
// ou
// os alunos em que tiverem o campo de email em seu document

13. Operações CRUD do MongoDB

13.3 Operações de leitura

operadores all / in

Esses operadores existem para trabalharmos com arrays.

O operador **\$all** retorna o documento se todos os itens procurados estiverem contidos na query. E o operador **\$in** varre o array mostrando os documents que contenham algum dos itens especificados.

```
>> db.comidas.find({ favoritas: {$all: ['hot dog', 'cerveja']} })
// retorna os documents que possuem hot dog e cerveja dentro do array de
favoritas
```

```
>> db.comidas.find({ favoritas: {$in: ['churrasco', 'Hamburguer']} })
// retorna os documents que possuem pelo menos um dos itens no array
passado
```

13. Operações CRUD do MongoDB

13.3 Operações de leitura

Você pode especificar filtros de consulta ou critérios que identificam os documentos a serem retornados.

```
db.users.find(  
  { age: { $gt: 18 } },  
  { name: 1, address: 1 }  
).limit(5)
```

- ← collection
- ← query criteria
- ← projection
- ← cursor modifier

13. Operações CRUD do MongoDB

13.4 Operações de atualização

As operações de atualização modificam os documentos existentes em uma coleção . O MongoDB fornece os seguintes métodos para atualizar documentos de uma coleção:

```
>> db.pessoas.update(  
    {nome: Luiza},  
    {nome: 'Luiza Magalu', english: true}  
)
```

A função update recebe dois objetos como parâmetro, o primeiro é a query para busca, e o segundo objeto é o replace, ou seja, são os dados que substituirão aquele document.

13. Operações CRUD do MongoDB

13.4 Operações de atualização

set

O operador set é usado quando queremos evitar que a substituição inteira do document aconteça, ou seja, quando queremos acrescentar e / ou alterar somente alguns campos no document, sem apagar todos os outros.

```
>> db.pessoas.update(  
    { nome: Luiza },  
    { $set: {idade:22} }  
)
```

13. Operações CRUD do MongoDB

13.4 Operações de atualização

upsert

Cria um novo document, caso não exista nenhum com as características passadas.

```
>> db.pessoas.update(  
    {nome: "Amanda"},  
    {$set: {idade: 40}}, {upsert: true}  
)
```

13. Operações CRUD do MongoDB

13.4 Operações de atualização

unset

Remove os campos especificados nele.

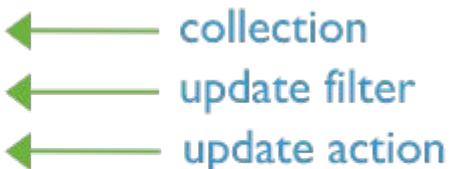
```
>> db.pessoas.update(
    { nome: 'seu madruga' },
    { $unset: {profissao:1} }
)
// remove o campo profissão do document onde o nome é seu madruga
```

13. Operações CRUD do MongoDB

13.4 Operações de atualização

Você pode especificar critérios, ou filtros, que identificam os documentos a serem atualizados. Esses filtros usam a mesma sintaxe das operações de leitura.

```
db.users.updateMany(  
  { age: { $lt: 18 } },  
  { $set: { status: "reject" } }  
)
```



13. Operações CRUD do MongoDB

13.4 Operações de atualização

Campos:

Nome	Descrição
\$currentDate	Define o valor de um campo para a data atual, como uma data ou um carimbo de data/hora.
\$inc	Incrementa o valor do campo pelo valor especificado.
\$min	Atualiza o campo apenas se o valor especificado for menor que o valor do campo existente.
\$max	Atualiza o campo apenas se o valor especificado for maior que o valor do campo existente.
\$mul	Multiplica o valor do campo pelo valor especificado.
\$rename	Renomeia um campo.
\$set	Define o valor de um campo em um documento.
\$setOnInsert	Define o valor de um campo se uma atualização resultar na inserção de um documento. Não tem efeito nas operações de atualização que modificam documentos existentes.
\$unset	Remove o campo especificado de um documento.

13. Operações CRUD do MongoDB

13.5 Operações de exclusão

As operações de exclusão removem documentos de uma coleção. O MongoDB fornece os seguintes métodos para excluir documentos de uma coleção:

```
db.collection.deleteOne()  
db.collection.deleteMany()
```

No MongoDB, as operações de exclusão visam uma única coleção . Todas as operações de gravação no MongoDB são atômicas no nível de um único documento.

Você pode especificar critérios, ou filtros, que identificam os documentos a serem removidos. Esses filtros usam a mesma sintaxe das operações de leitura.

13. Operações CRUD do MongoDB

13.5 Operações de exclusão

```
db.users.deleteMany(  
  { status: "reject" }  
)
```



collection
delete filter

14. Agregação no MongoDB

14. Agregação no MongoDB

14.1 Agregação

As operações de agregação processam vários documentos e retornam os resultados calculados. Você pode usar operações de agregação para:

- **Agrupe valores** de vários documentos juntos.
- **Execute operações** nos **dados agrupados** para retornar um único resultado.
- **Analise** as **mudanças** de dados ao longo do tempo.

14. Agregação no MongoDB

14.2 Pipelines de agregação

Um pipeline de agregação consiste em um ou **mais estágios** que **processam documentos**:

- Cada estágio **realiza uma operação** nos documentos de entrada. Por exemplo, um estágio pode filtrar documentos, agrupar documentos e calcular valores.
- Os documentos que saem de um estágio são passados para o próximo estágio.
- Um pipeline de agregação pode retornar resultados para grupos de documentos. Por exemplo, retorne os valores total, médio, máximo e mínimo.

14. Agregação no MongoDB

14.3 Exemplo de pipeline de agregação

O exemplo de pipeline de agregação a seguir contém dois estágios e retorna a quantidade total do pedido de pizzas de tamanho médio agrupadas por nome de pizza:

```
db.orders.aggregate( [  
    // Stage 1: Filter pizza order documents by pizza size  
    {  
        $match: { size: "medium" }  
    },  
  
    // Stage 2: Group remaining documents by pizza name and calculate  
    // total quantity  
    {  
        $group: { _id: "$name", totalQuantity: { $sum: "$quantity" } }  
    }  
] )
```

14. Agregação no MongoDB

14.3 Exemplo de pipeline de agregação

O estágio **\$match**:

Filtrar os **documentos** do **pedido de pizza** para **pizzas size** com **medium**.

Passa os demais documentos para o estágio **\$group**.

O estágio **\$group**:

Agrupa os **documentos restantes** por **pizza name**.

Usa **\$sum** para calcular o **pedido total quantity** de cada **pizza name**. O total é armazenado no **totalQuantitycampo** retornado pelo pipeline de agregação.

14. Agregação no MongoDB

14.4 Métodos de agregação de propósito único

Você pode usar os seguintes métodos de agregação de propósito único para agrregar documentos de uma única coleção:

Método	Descrição
<code>db.collection.estimatedDocumentCount()</code>	Retorna uma contagem aproximada dos documentos em uma coleção ou exibição.
<code>db.collection.count()</code>	Retorna uma contagem do número de documentos em uma coleção ou exibição.
<code>db.collection.distinct()</code>	Retorna uma matriz de documentos que possuem valores distintos para o campo especificado.

15. Mapeamento SQL para MongoDB

15. Mapeamento SQL para MongoDB

15.1 Terminologia e Conceitos

A tabela a seguir apresenta as várias terminologias e conceitos SQL e a terminologia e conceitos correspondentes do MongoDB.

Termos/Conceitos SQL	Termos/Conceitos do MongoDB
base de dados	base de dados
tabela	coleção
fila	documento
coluna	campo
índice	índice
aggregation (group by)	aggregation pipeline

Perguntas?



magalu



#VemSerFeliz