

Banco de Dados e Introdução a SQL

Labenu_



O que vamos ver hoje?

- Tipo de Banco de Dados
(Relacional x Não Relacional)
- SQL (Criar / Deletar / Inserir Valores / Remover Valores/ Consultar valores)



Relembrando...

Labenu_



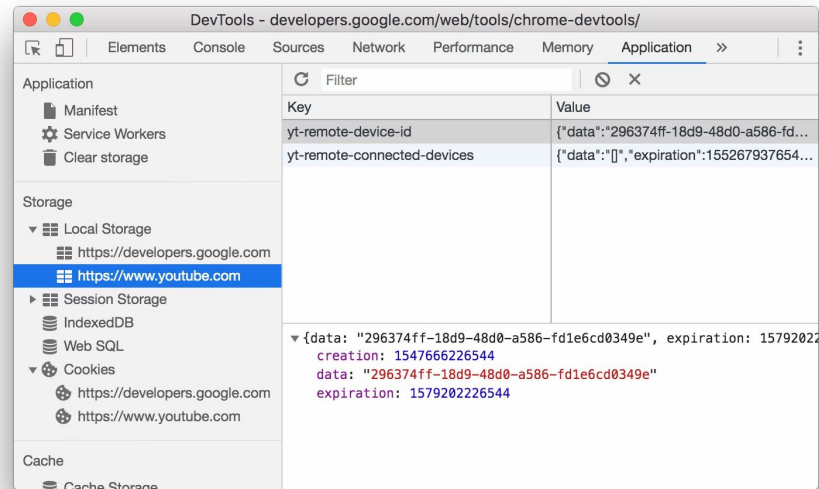
Persistindo os Dados no Front

- No front-end para persistirmos os dados, utilizamos o Local Storage.

Inspecionar > Application > Storage > Local Storage

`localStorage.getItem ()` => Pega o que está guardado

`localStorage.setItem ()` => Guarda as informações



Tipos de Banco de Datos

Labenu_



Não Relacional (NoSQL)



Relacional (SQL)



- Dados estruturados em documentos
- **Não possuem estrutura definida**
- **Organização e formas de acesso não padronizados**
- Varia bastante de cada banco

- Dados **divididos em tabelas**, com *schema* (estrutura) definido
- Tabelas podem possuir **relações**
- Organização **padronizada**
- Utilizam **linguagem SQL** (*Structured Query Language*)



Introdução à SQL

Labenu_



Persistência no Backend

Até agora, recorreremos aos **armazenamentos em memória** ou ao armazenamento em arquivos de texto (ex: JSON)

- **Bancos de Dados**

- Relacional (tabelas e ids)
- Linguagem SQL
- Gerenciador de banco de dados MySQL
- Aplicativos mySQL, BeeKeeper ou extensões VSCode



Exemplo de tabela

Cada **tabela** é uma entidade

Pessoas			
id (PK)	nome	email	idade
1	Fulano	fulano@gmail.com	35
2	Ciclana	ciclana@gmail.com	24
3	Alice	alice@gmail.com	33
4	Bob	bob@gmail.com	52

Linha, são os registros de uma tabela

Colunas são os campos(parâmetros) de uma tabela



Structure Query Language (SQL)

- **Linguagem de Consulta Estruturada**

é a linguagem padrão de banco de dados Relacionais

- É amplamente utilizada no mercado

- Baixa curva de aprendizado

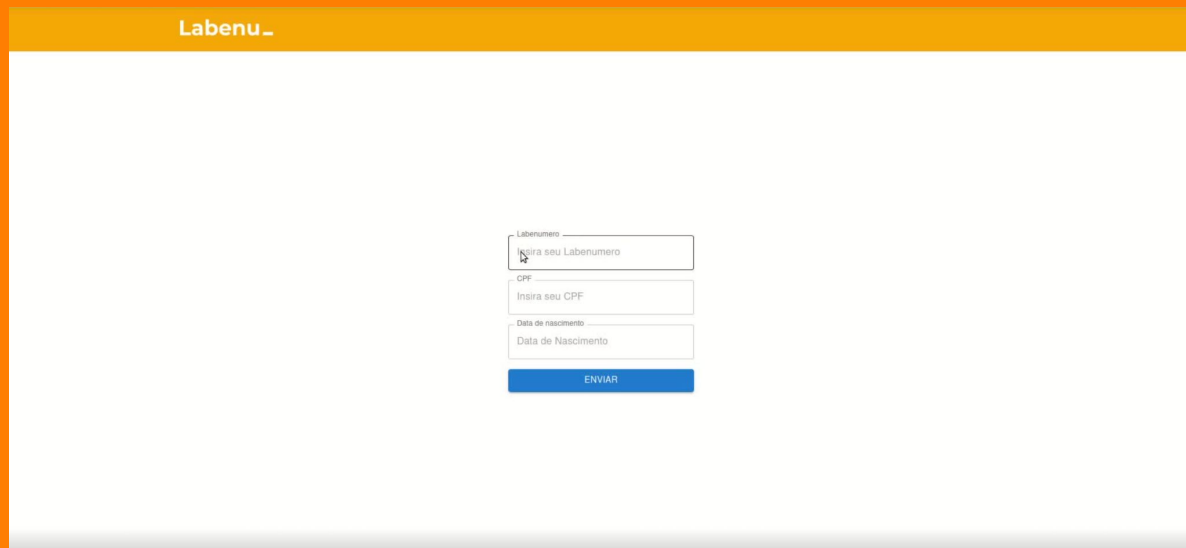


Conhecendo o Workbench

Labenu_



Crie o seu banco!



The screenshot shows a web form titled "Labenu_" in a yellow header bar. The form itself is white and contains four input fields stacked vertically, each with a label above it: "Labenumero", "CPF", "Data de nascimento", and "Data de Nascimento". Below these fields is a blue button labeled "ENVIAR". A mouse cursor is visible over the "Labenumero" input field.

⚠ Importante ⚠ : Guarde os seus dados, pois é difícil essa recuperação dos dados!

Labenu_

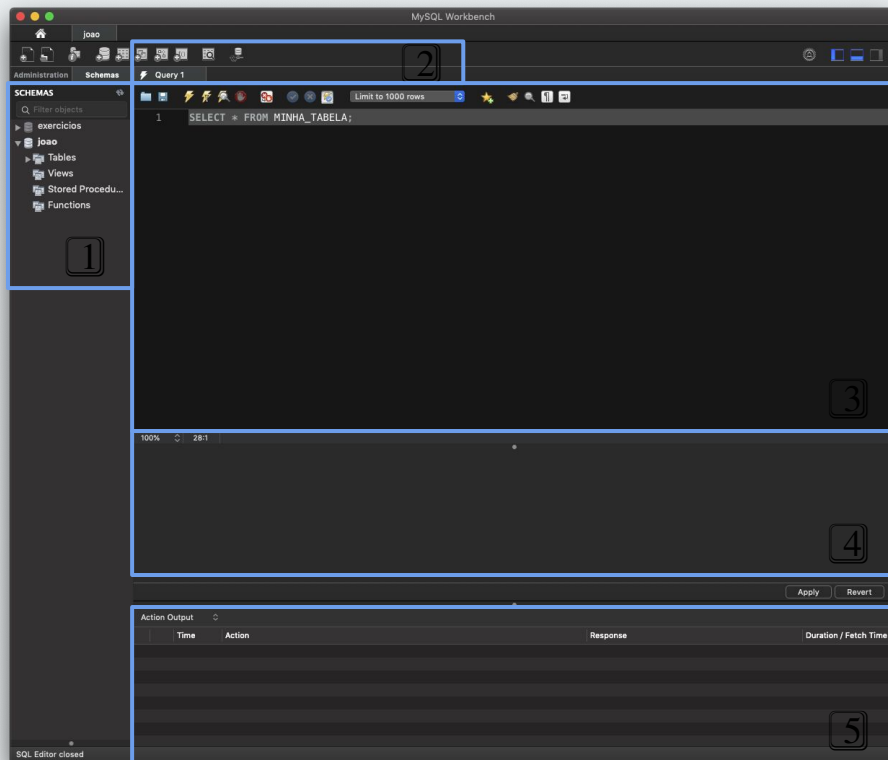


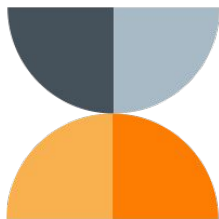
Conhecendo o Workbench

- As partes mais importantes da interface do Workbench são as destacadas ao lado.

Temos:

- Schemas **1**
- Navegador de queries **2**
- Editor de queries **3**
- Visualização de tabelas **4**
- Visualizador de resultados **5**





- Bancos de dados são programas que gerenciam o armazenamento, escrita e leitura de dados de forma eficiente
- **Bancos relacionais** são um tipo desses programas
- Dados são organizados em **tabelas**, que possuem **itens** com **campos**
- **Chave primária** é um identificador que todo item deve ter



Criando e Deletando Tabelas

Labenu_



Tipos principais de Dados

- **INT** - Números inteiros
- **DOUBLE** - Números com ponto flutuante
- **VARCHAR(n)** - String com no máximo N caracteres
- **CHAR** - String com 1 caractere
- **TEXT** - String com quantidade quase ilimitada de caracteres
- **ENUM** - Objeto com strings pré-definidas
- **DATE** - Representa data (YYYY-MM-DD)
- **DATETIME** - Representa data e tempo (YYYY-MM-DD hh:mm:ss)

Para saber mais, consulte: [SQL Data Types](#)



Restrições principais

- **PRIMARY KEY** - Chave primária (chave única na tabela)
- **FOREIGN KEY** - Chave estrangeira (chave única na tabela)
- **NULL / NOT NULL** - Indica se a coluna pode ser ou não pode ser nula
- **UNIQUE** - Indica que o valor deve ser único
- **AUTO_INCREMENT** - Indica que o valor é auto incrementável
- **DEFAULT** - Define um valor padrão caso nenhum valor seja passado

Para saber mais, consulte: [SQL Restrições](#)



SINTAXE - Criar Tabela

```
CREATE TABLE nome_tabela (  
  coluna1 INT PRIMARY KEY,  
  coluna2 VARCHAR(255) UNIQUE,  
  coluna3 CHAR NULL DEFAULT 'X',  
  coluna4 ENUM('op1', 'op2') NOT NULL,  
  ....  
);
```



Comando



Tipo da Variável



Restrição





Exercício 1



```
CREATE TABLE Pessoas (  
  id          INT          PRIMARY KEY,  
  nome        VARCHAR(255) NOT NULL,  
  email       VARCHAR(255) NOT NULL UNIQUE,  
  idade       INT          DEFAULT 18  
);
```



Comando



Tipo da Variável



Restrição

Resultado:

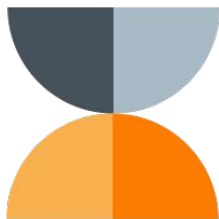
Pessoas			
id	nome	email	idade

SINTAXE - Deletando Tabela

- Usamos o comando **DROP TABLE** para deletar as tabelas (enquanto não temos relações entre as tabelas... Cenas das próximas aulas)
- O comando recebe o nome da tabela

```
DROP TABLE nome_da_tabela;
```





Pausa para relaxar 🤪

- No backend, armazenamos os dados em **bancos**
- O banco relacional é formado por tabelas
- Toda tabela deve ter uma **chave primária** [PK]
- Os atributos das tabelas possuem tipos definidos
- Para criar uma tabela usamos

CREATE TABLE nome_da_tabela



Inserindo e Removendo Valores

Labenu_



SINTAXE - Inserindo Valores na Tabela

- Usamos o comando **INSERT** para adicionar valores na tabela
- O comando recebe:
 - O nome da tabela
 - Os nomes das colunas em ordem de inserção
 - Os valores

```
INSERT INTO nome_tabela (coluna1, coluna2, coluna3, ...)
    VALUES (valor1, valor2, valor3, ...);
```





Exercício 2



```
INSERT INTO Pessoas (id, nome, email, idade)  
VALUES (1, "Fulano", "fulano@gmail.com", 35),  
        (2, "Ciclana", "ciclana@gmail.com", 24);
```

Resultado:

Pessoas			
id	nome	email	idade
1	Fulano	fulano@gmail.com	27
2	Ciclana	ciclana@gmail.com	24

SINTAXE - Remover Valores na Tabela



- Usamos o comando **DELETE** para apagar uma ou mais linhas de uma tabela.
- O comando recebe:
 - O nome da tabela
 - Opcional: A condição **WHERE**
 - Se houver, apagará somente as linhas que atendam às condições.
 - Se **não** houver, **apagará todas as linhas** da tabela. ⚠



SINTAXE - Remover Valores na Tabela



Deleção segura (RECOMENDADA)

```
DELETE FROM nome_tabela  
WHERE condicao = valor;
```

Apaga todas as linhas da tabela, não apaga a tabela em si

```
DELETE FROM nome_tabela;
```





Exercício 3



```
DELETE FROM Pessoas  
WHERE id = 1;
```

Resultado:

Pessoas			
id	nome	email	idade
2	Ciclana	ciclana@gmail.com	24

Consultando valores na tabela

Labenu_



Consultando os valores na tabela

- Usamos o comando **SELECT** para consultar os valores
- O comando recebe:
 - Os nomes das colunas que serão buscadas
 - Para buscar todas as colunas, usa-se: * (asterisco)
 - O nome da tabela
 - Opcional: A condição **WHERE**
 - Se houver, retorna as linhas que atendem a condição
 - Se não houver, todas as linhas são retornadas



SINTAXE - Consultar Valores na Tabela



Retorna todos os valores de todas as colunas da tabela

```
SELECT * FROM nome_tabela;
```

Retorna os valores que atendam a condição das colunas especificadas

```
SELECT coluna1, coluna2 FROM nome_tabela  
WHERE condicao = valor;
```





Exercício 1

```
SELECT * FROM Pessoas ;
```

Resultado:

Pessoas			
id (PK)	nome	email	idade
1	Fulano	fulano@gmail.com	35
2	Ciclana	ciclana@gmail.com	24
3	Alice	alice@gmail.com	33
4	Bob	bob@gmail.com	52



Exercício 5



```
SELECT email, idade  
FROM Pessoas  
WHERE nome="Ciclana";
```

Resultado:

Resultado	
email	idade
ciclana@gmail.com	24



Exercício 6



Podemos renomear a coluna com um ALIAS (AS)

```
SELECT email, idade AS idade_em_anos  
FROM Pessoas  
WHERE nome="Ciclana";
```

Resultado:

Resultado	
email	idade_em_anos
ciclana@gmail.com	35



Exercício 7



Use esses exemplos, ou solte a criatividade para testar novos!

```
SELECT *  
FROM Pessoas  
WHERE idade = 24 OR nome = "Fulano";
```

```
SELECT id, nome  
FROM Pessoas  
WHERE nome LIKE "%an%";
```

```
SELECT email  
FROM Pessoas  
WHERE nome LIKE "%an%" AND  
idade > 30;
```

Extra: Visualizar a estrutura da tabela

- Usamos o comando **DESCRIBE** para conferir a estrutura da tabela e os tipos de dados de cada coluna

DESCRIBE Pessoas;

Resultado:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NUL	
nome	varchar(255)	NO		NUL	
email	varchar(255)	NO	UNI	NUL	
idade	int(11)	YES		NUL	



Dúvidas? 🧐

Labenu_





Obrigado!