Mãos a obra

Preparando o ambiente

➔ Vamos usar o **XAMPP** : pacote com os principais servidores de código aberto

- **X** - Disponível para várias plataformas (Windows, Linux e OS X)
- **A** - Apache (Servidor Web)
- **M** - MariaDB (Banco de dados)
- **P** - PHP
- **P** - Perl

Link para download:https://www.apachefriends.org/download.html

phpMyAdmin

- Ferramenta de software livre escrita em PHP
- Usada para administrar o MySQL/MariaDB na Web
  - Muito amigável com o usuário

Para acessá-lo: Abra o Xampp > Inicie o Apache e o MySQL > clique em Admin referente ao MySQL

Documentação: https://docs.phpmyadmin.net/en/latest/

MySQL Workbench

➔ MySQL Workbench oferece ferramentas visuais para criar, executar e otimizar consultas SQL.

Download: https://www.mysql.com/products/workbench/

MariaDB (“Irmão” do MySQL?)

- Criado por Michael Monty Widenius (2009)
  - Criador do MySQL (1995)
- Algumas **vantagens** com relação ao MySQL:
  - Velocidade;
  - Menor exigência de hardware;
  - Total compatibilidade com o MySQL.

Documentação: https://mariadb.org/documentation/

SENAI

SQL

Structured Query Language

SQL - Structured Query Language

- Criador Dr. E. F. Codd (1970)
- Linguagem de programação para **armazenar** e **processar informações** em um banco de dados relacional
- Instruções SQL são úteis para:
  - Armazenar
  - Atualizar
  - Remover
  - Pesquisar e recuperar informações

Comandos SQL

Linguagem de definição de dados (Data Definition Language - DDL)

Comandos SQL que projetam a estrutura do banco de dados

- Create (Criar)
- Alter (Alterar)
- Drop (Deletar)

Tipos de dados numéricos (MySQL/MariaDB)

- **INT** – número inteiro de tamanho comum (variações para tamanho TINYINT, SMALLINT, MEDIUMINT e BIGINT)
- **DECIMAL** – número decimal, de ponto fixo;
- **FLOAT** – número de ponto flutuante de precisão simples (32 bits);
- **DOUBLE** – número de ponto flutuante de precisão dupla (64 bits);
- **BIT** – um campo de um bit (tem tamanho 1 byte)

Tipos de dados string (MySQL/MariaDB)

- **CHAR** – uma cadeia de caracteres (string), de tamanho fixo.
- **VARCHAR** – uma string de tamanho variável e não-binária;
- **BLOB** – um BLOB (Binary Large OBject – OBjeto Grande Binário) usado para guardar imagens, por exemplo;
- **TEXT** – uma cadeia de caracteres grande com tamanho fixo de 65,535 caracteres

Tipos de dados data e hora (MySQL/MariaDB)

- **DATE -** Guarda data no formato 'YYYY-MM-DD'
- **DATETIME -** Guarda data e hora no formato YYYY-MM-DD HH:MM:SS.ffffff com limites entre 1000-01-01 00:00:00.000000 and 9999-12-31 23:59:59.999999.
- **YEAR -** Guarda ano no formato YYYY

https://dev.mysql.com/doc/refman/8.0/en/numeric-types.html

Criando um banco de dados

➔ Comando básico para criar um banco de dados

```
CREATE database nome_bd;
```

➔ Comando para usar um banco de dados

```
USE nome_bd;
```

O “;” é importante no script!

CREATE TABLE

```
CREATE TABLE nome_tabela (campo1 tipo_campo1, campo2 tipo_campo2, … ,campon tipo_campon)
```

Exemplo:

```
CREATE TABLE Pessoa(
    codPessoa int,
    Sobrenome varchar(255),
    Nome varchar(255),
    DataNasc datetime
);
```

O comando **describe** nome_tabela apresenta a descrição da tabela nome_tabela

CREATE TABLE (Atributos Opcionais)

```
CREATE TABLE nome_tabela (campo1 tipo_campo1  NULL/
NOT NULL, campo2 tipo_campo2 NULL/NOT NULL, … ,campon
tipo_campon NULL/NOT NULL)
```

```
CREATE TABLE Pessoa(
    codPessoa int NOT NULL,
    Sobrenome varchar(255) NOT NULL,
    Nome varchar(255),
    DataNasc datetime
);
```

CREATE TABLE (com chave primária)

```
CREATE TABLE nome_tabela (nome_coluna1 tipo_coluna1,
nome_coluna2, tipo_coluna2, … , nome_campon
tipo_colunan, PRIMARY KEY (nome_colunak))
```

```
CREATE TABLE Pessoa(
    codPessoa int NOT NULL,
    sobrenome varchar(255) NOT NULL,
    nome varchar(255),
    dataNasc datetime,
     PRIMARY KEY (codPessoa)
);
```

Se for composta basta adicionar vírgula. Ex: PRIMARY KEY (codPessoa, sobrenome)

CREATE TABLE (com chave estrangeira)

```
CREATE TABLE nome_tabela (nome_coluna1 tipo_coluna1,
nome_coluna2, tipo_coluna2, … , nome_campon tipo_colunan,
campon tipo_campon, FOREIGN KEY (nome_coluna_fk) REFERENCES
nome_tabela_referente(campo_na_tabelareferente))
```

```
CREATE TABLE Pedido(
    codPedido int,
    numeroPedido int,
    codCliente int,
    PRIMARY KEY (codPedido),
    FOREIGN KEY (codCliente) REFERENCES Cliente(codCliente)
);
```

ALTER TABLE

➔ Adicionando coluna a uma tabela existente

```
ALTER TABLE nome_tabela
ADD nome_coluna tipo_coluna;
```

➔ Removendo coluna de uma tabela existente

```
ALTER TABLE nome_tabela
DROP COLUMN nome_coluna;
```

ALTER TABLE

➔ Alterando o tipo de dado de uma coluna existente

```
ALTER TABLE nome_tabela
MODIFY COLUMN nome_coluna tipo_coluna_novo;
```

ALTER TABLE

➔ Altera o nome e tipo do dado da coluna de uma tabela

```
ALTER TABLE nome_tabela
CHANGE COLUMN nome_antigo nome_novo tipo_novo;
```

Também podemos alterar a obrigatoriedade, basta adicionar o NOT NULL

ALTER TABLE

➔ Adiciona uma chave primária

```
ALTER TABLE nome_tabela
ADD CONSTRAINT PRIMARY (id_tabela);
```

➔ Remove chave primária

```
ALTER TABLE nome_tabela
DROP PRIMARY KEY;
```

ALTER TABLE

➔ Adiciona uma chave estrangeira

```
ALTER TABLE nome_tabela

ADD CONSTRAINT FOREIGN KEY (coluna_id) REFERENCES
nome_tabela2 (coluna_id);
```

➔ Remove uma chave estrangeira

```
ALTER TABLE nome_tabela

DROP FOREIGN KEY nome_da_chave_estrangeira;
```

DROP TABLE

➔ Remover uma tabela existente

```
DROP TABLE nome_tabela
```

➔ Remover um banco de dados existente

```
DROP DATABASE nome_tabela
```

➔ Deleta os registros da tabela

```
TRUNCATE TABLE nome_tabela
```

RENAME TABLE

➔ Renomear uma tabela existente

```
RENAME TABLE nome_tabela TO nome_tabela_novo
```