MÓDULO

# PROGRAMAÇÃO PHP

UNIDADE

## PHP E MYSQL



Instituto Tecnológico

# ÍNDICE

# OBJETIVOS

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Criar uma base de dados simples em MySQL.
- Conectar o PHP à base de dados MySQL.
- Ler dados de tabelas MySQL de PHP.
- Inserir dados nas tabelas MySQL.
- Modificar os dados do MySQL.

# INTRODUÇÃO

Nesta unidade verá como interagir com uma base de dados de PHP. Em geral, o PHP está integrado com o sistema de base de dados MySQL, do qual verá as funções mais comuns para poder fazer uma página web que se conecte à base de dados e exiba os seus dados.

Aprenderá também a inserir e modificar os registos das tabelas da base de dados MySQL.

# 1. MYSQL

O MySQL é um sistema de gestão de base de dados relacional que permite ter os dados numa estrutura dividida por tabelas que contêm os campos de identificação e os valores de cada registo.

A vantagem dos gestores de base de dados sobre os ficheiros XML, JSON ou texto simples é que estes permitem criar um sistema de índices nas tabelas, ou seja, a busca por um determinado registo é consideravelmente reduzida.

Também facilita a pesquisa em várias tabelas, pois pode criar consultas SQL (linguagem do gestor de base de dados), que incluem várias tabelas e até mesmo consultas que usam tabelas para determinar quais os registos de outra tabela utilizada.

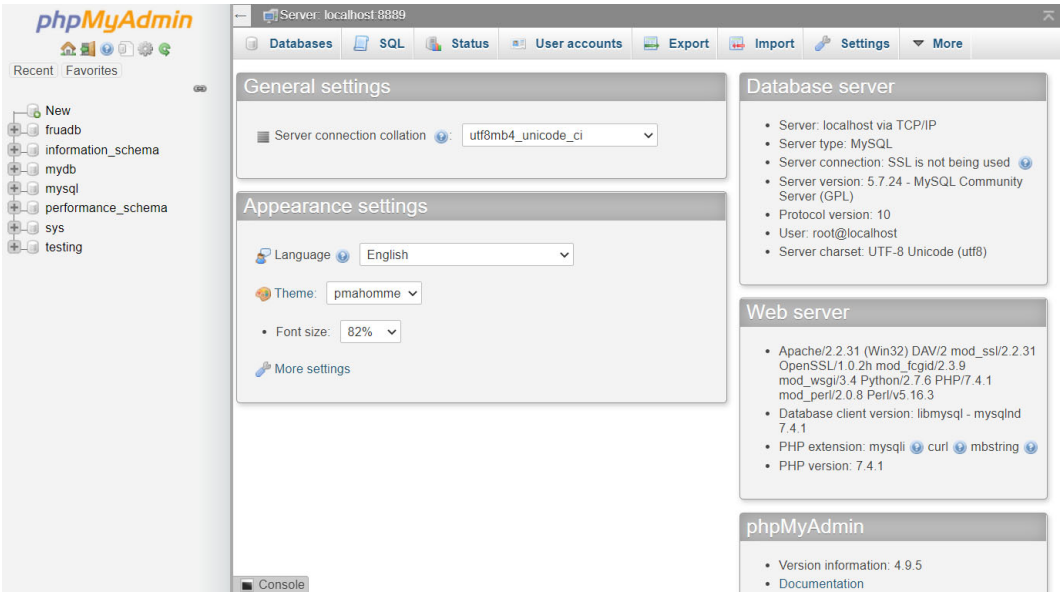
A linguagem SQL do MySQL é muito extensa, logo, aprenderá apenas o essencial para poder trabalhar de forma eficiente com as bases de dados.

## O gestor MySQL

A primeira coisa que deve ter instalado é o gestor MySQL e o próprio mecanismo MySQL. Se instalou o pacote para Windows, com as opções padrão selecionadas, que são escolhidas durante a instalação, o software deve estar instalado corretamente. Para verificar, basta tentar abrir o seguinte link (necessita do servidor web):

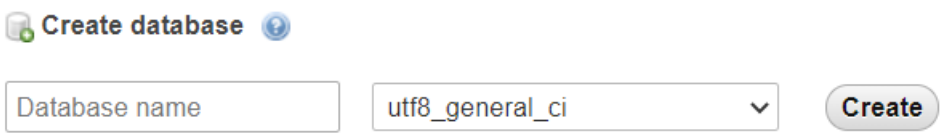
- http://localhost/phpMyAdmin/

Ser-lhe-ão solicitados o utilizador e a palavra-passe de acesso: o utilizador é "root" e a palavra-passe é a que foi escolhida durante a instalação. Depois de entrar, encontrará a seguinte janela:




Ecrã principal de gestão do MySQL.

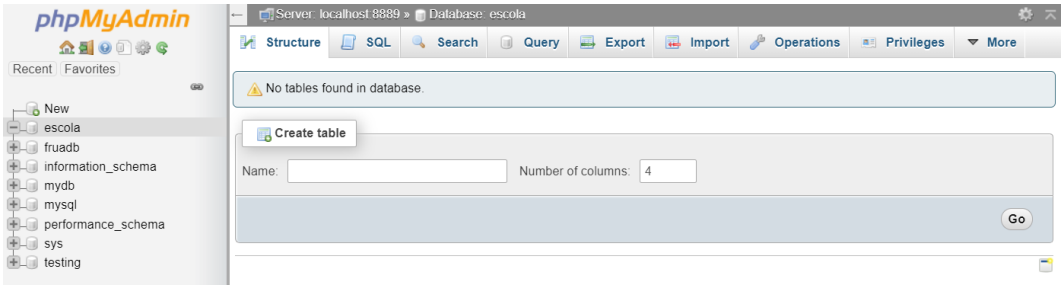
Tem a opção de utilizar o MySQL em português, mas geralmente é utilizado em inglês. No lado esquerdo do ecrã estão as bases de dados disponíveis. O que verá no seu ecrã são as bases de dados criadas por defeito. Para criar uma nova base de dados basta clicar na opção "New".



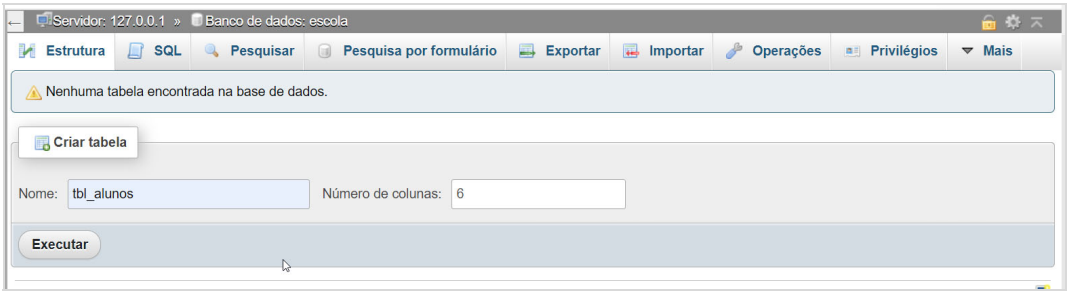

Opção no ecrã principal para criar uma nova base de dados.

No primeiro campo é colocado o nome da base de dados sem espaços. Para esta unidade será criada uma base de dados chamada "escola". No segundo campo é utilizado **utf8_unicode_ci**, como mostra a imagem (foi escolhido porque nas páginas web de exemplo é utilizado **utf8** como o codec da web e, portanto, não terá de fazer transformações para caracteres).

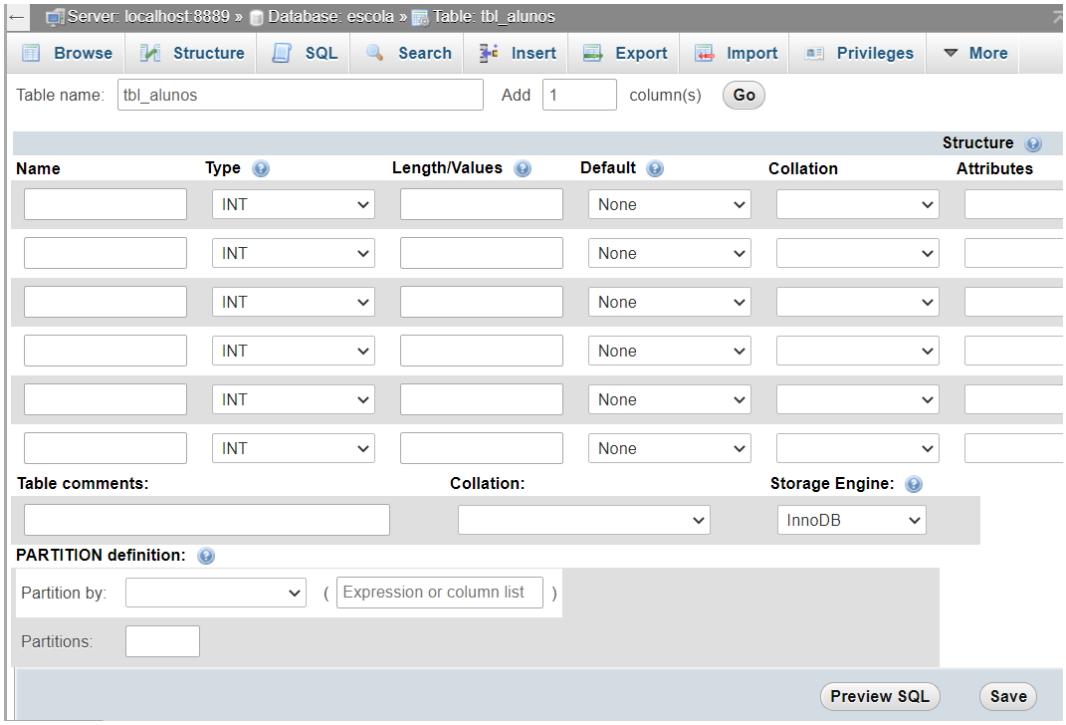
Se tudo estiver correto, deverá aparecer uma mensagem com um aviso, e passará à janela de criação de tabela:




Na parte inferior é pedido um nome para a tabela e o número de campos que esta terá (este número pode ser modificado posteriormente). O nome da tabela não pode conter espaços e deve começar com um carácter ou underscore. Geralmente, é escolhido um tipo de nomenclatura para organizar as tabelas. Neste exemplo, será utilizado o prefixo **tbl**, logo, o nome da tabela será, por exemplo, "tbl_alunos" e seis campos.




Ao clicar no botão "Go/Executar", avançar-se-á para uma janela na qual serão inseridos os campos da tabela:




Janela de criação de campos da tabela.

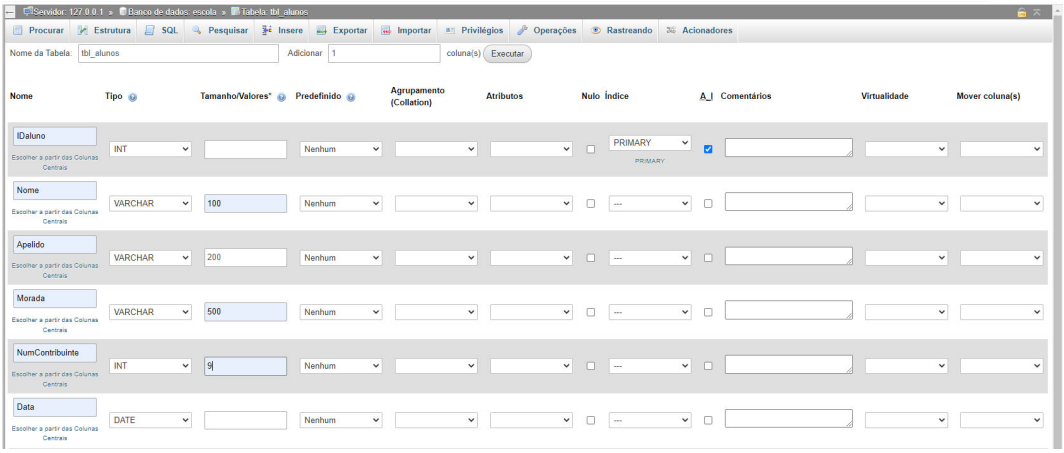
Nesta janela existem vários campos para preencher:

- **Name**: que irá conter o nome que se pretende dar ao campo.
- **Type**: de acordo com o tipo de dados que se pretende que contenha:
  - varchar: para strings.
  - int: para números inteiros.
  - float: para números decimais.
  - bool: para campos "sim/não".
  - text: para campos de texto longo (varchar aceita um número limitado de caracteres, dependendo da versão do MySQL).
  - date: para datas.
  - Os restantes são menos utilizados.
- **Lenght/Value**: se se pretender definir os campos de comprimentos exatos como seria, por exemplo, o número de contribuinte.
- **Default**: utilizado para atribuir um valor padrão ao campo e evitar que este fique vazio.

- **Collation**: regra geral, não será alterado e ficará com o valor selecionado ao criar a tabela.
- **Attributes**: não será alterado, por agora.
- **Null**: indica se o campo introduzido pode estar vazio ou se é obrigatório que tenha um valor.
- **Index**: indica se o campo em questão é um índice e que tipo de índice.
    - Primary: para definir o campo como a chave da tabela (geralmente é um número autoincremental).
    - Unique: para indicar que o valor do campo deve ser único na tabela, ou seja, o mesmo valor não pode ser repetido, por exemplo, o campo "e-mail", se não pretender que existam vários registos com o mesmo e-mail. O campo-chave da tabela é, por padrão, exclusivo.
    - Index: para gerir um índice para o campo. É bastante útil marcá-lo, em bases de dados muito grandes, nos campos que são utilizados para pesquisa.
    - Os restantes são menos utilizados e de momento não serão abordados.
- **A_I,** (Auto Increment) permite selecionar se o campo será autoincrementado, ou seja, não será necessário definir um valor para preenchê-lo com o próximo valor livre. Esta opção é geralmente utilizada no campo-chave da tabela para que não haja repetições.
- Os restantes são menos utilizados.

Como exemplo, foram criados os seguintes campos:

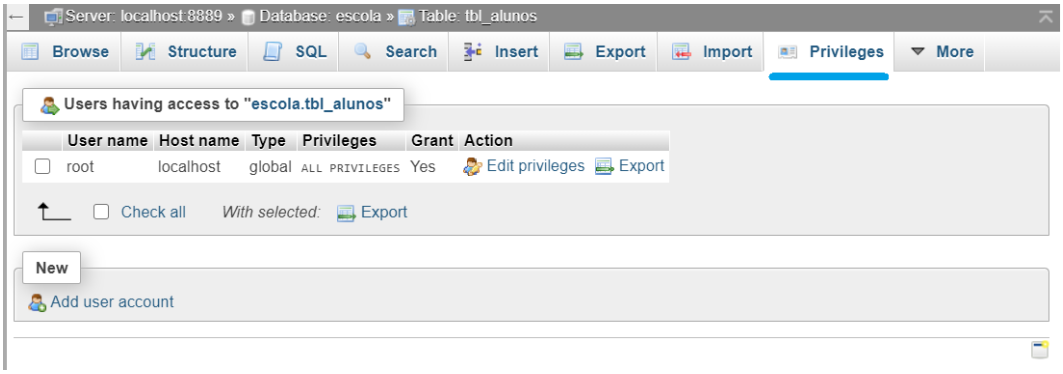
- IDaluno, INT, autoincrement, Primary.
- Nome, varchar, 100.
- Apelido, varchar, 200.
- Morada, varchar, 500.
- NumContribuinte, INT, 9.
- Data, date.




No final da criação da tabela, aparecerá uma janela que informa se a tabela foi criada corretamente e permite modificar os campos.


No desenvolvimento normal, deve saber-se quais as opções que o utilizador da base de dados vai desenvolver, ou seja, se a partir da página web irá apenas ler dados, ler e escrever, etc. Com estas informações será criado um utilizador da base de dados com as permissões necessárias. Para simplificar o processo nesta unidade, será utilizado o utilizador **root**, que é o administrador e tem acesso a todas as permissões.

Para criar um utilizador, acede-se à aba “Privileges” (Privilégios) do ecrã inicial:



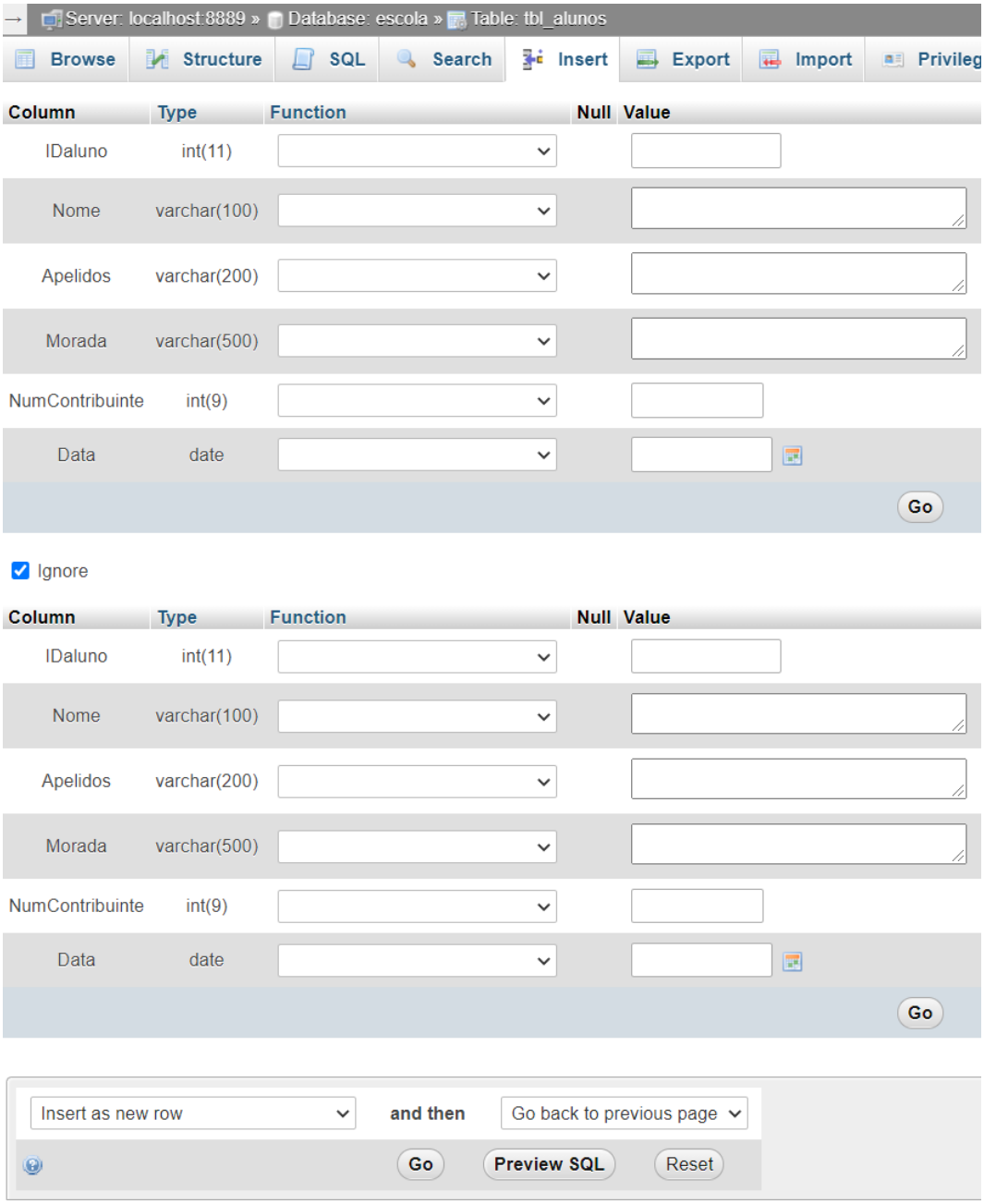

Aba do ecrã inicial a partir da qual se criam utilizadores.

Voltando à tabela "tbl_alunos" criada anteriormente, aparece um menu na parte superior com várias abas:

- **Browse (Procura)**: permite ver o conteúdo da tabela (os seus registos), editá-lo ou excluí-lo.
- **Structure (Estrutura)**: permite modificar a estrutura da tabela, adicionar ou remover campos, alterar o tipo de dados, etc.
- **Insert (Inserir)**, permite incluir novos registos na tabela.
- As restantes abas não serão abordadas, inicialmente.


Principais abas do menu de uma tabela.

A janela para inserir os dados é a seguinte:




Janela de inserção de dados.

Na parte superior são mostrados os campos que devem ser preenchidos. Em seguida, aparece uma caixa de seleção com o nome “ignore” e, novamente, todos os campos (esta secção será ignorada); no final, uma vez que os campos superiores tenham sido preenchidos, pressiona-se o botão “Go” para inserir o novo registo.

Com estas informações fica a saber praticamente tudo o que é necessário sobre o MySQL para poder desenvolver uma base de dados para as páginas web.

# 2. CONEXÃO DA BASE DE DADOS COM PHP

O PHP, tal como as outras linguagens de programação, pode conectar-se a muitos tipos de bases de dados. Possui várias bibliotecas de conexão nativas que permitem a conexão a um driver específico (aumentando assim a velocidade) para as seguintes bases de dados:

- CUBRID.
- DB++.
- dBase.
- filePro.
- Firebird/InterBase.
- FrontBase.
- IBM DB2.
- Informix.
- Ingres.
- MaxDB.
- Mongo.
- MongoDB.
- mSQL.
- Mssql.
- MySQL.

- OCI8.
- Paradox.
- PostgreSQL.
- SQLite.
- SQLite3.
- SQLSRV – driver de Microsoft SQL Server para PHP.
- Sybase.
- tokyo_tyrant.

Para cada uma destas bases de dados existe um conjunto de funções que permitem conectar e trabalhar diretamente com as bases de dados. No entanto, isso não significa que algumas bases de dados, como o SQL Server (que não estão na lista) não estejam acessíveis; o PHP simplesmente utiliza um driver de conexão direta, chamado ODBC, que permite aceder a essas bases de dados que não possuem um driver específico.

No final da unidade, nas Propostas de Desenvolvimento do Estudo, existem dois links onde pode ver todos os conjuntos de funções tanto do ODBC quanto das restantes bases de dados da lista; neste caso, aprenderá a usar somente os do MySQL, por ser o mais utilizado em conjunto com o PHP.

# 3. PHP MYSQL

Vão ser abordadas uma a uma as funções necessárias para aceder ao MySQL através do PHP, e assim carregar dados de forma dinâmica na página web ou recolher dados do utilizador na base de dados.

- **mysqli_connect**: a primeira coisa a fazer para obter dados de uma tabela MySQL do PHP é conectar a página web à base de dados. Para isso, utiliza-se a função mysqli_connect, para a qual serão passados, obrigatoriamente, quatro parâmetros:
  - Servidor: o nome do servidor MySQL ou o IP, se não se souber o nome (neste exemplo será "localhost").
  - Utilizador: o nome dos utilizadores da base de dados (neste caso, "root").
  - Senha: a senha de acesso do utilizador (a mesma que foi inserida na instalação).

  Esta função devolve o identificador da conexão à base de dados ou, em caso de erro, devolve false.

```
<?php
$conn = mysqli_connect('localhost', 'root', 'senha', 'escola');
if (!$conn) {
echo ('Não foi possível conectar: '. mysqli_connect_error());
}
?>
```

- **mysqli_connect_error**: esta função devolve o erro gerado pela conexão MySQL em formato de string. No exemplo anterior, poderá ver a sua utilização.
- **mysqli_select_db**: a etapa seguinte é selecionar a base de dados que será utilizada dentro do MySQL (neste caso, "escola"), que será o segundo parâmetro da função. Como primeiro parâmetro é passado o identificador da conexão MySQL.

```
$baseDados = mysqli_select_db($conn, 'escola');
```

Este processo devolverá true, se nenhum erro tiver surgido, caso contrário, devolverá false.

- **mysqli_close**: esta função fecha a conexão com o MySQL. Para isso, deverá passar o identificador da base de dados como um parâmetro (obtido através do mysqli_connect).

```
mysqli_close ($conn);
```

- **mysqli_query**: com esta função são enviadas as instruções para o MySQL. Como primeiro parâmetro recebe o identificador da conexão e como segundo parâmetro a instrução.

  A função devolve um ou outro tipo de valor, de acordo com a instrução enviada:

  - Para o SELECT, SHOW, DESCRIBE e outros que devolvem um conjunto de valores, a função devolve um resource, que é um tipo específico para elementos carregados de fora do PHP, mas que podem passar para um array.
  - Para os outros tipos de instruções, como o INSERT, UPDATE, DELETE, etc., devolve true, se nenhum problema tiver surgido, e false, caso contrário.

  Seguem-se exemplos de como utilizar as principais instruções SQL:

  - SELECT: esta instrução é utilizada para ler de uma ou mais tabelas MySQL e a sua sintaxe é: SELECT [os campos separados por vírgulas] FROM [tabela] WHERE [opcional: as condições seriam criadas aqui] ORDER BY [opcional: campos separados por vírgulas e com o sufixo asc ou desc].

    Exemplos de SELECT na tabela "tbl_alunos":

```
SELECT nome, apelidos FROM tbl_alunos;
```

Devolve os campos nome e apelidos de todos os alunos da tabela.

```
SELECT nome,apelidos FROM tbl_alunos WHERE nome = ‘Carlos’;
```

Irá devolver o nome e apelido de todos os alunos da tabela cujo nome é “Carlos”.

```
SELECT * FROM tbl_alunos WHERE YEAR(data) >= ‘2001’
```

Devolve todos os campos, utilizando um asterisco (*), da tabela “tbl_alunos”, quando a data do registo for 2001 ou posterior.

```
SELECT * from tbl_alunos ORDER BY nome asc
```

Devolve todos os registos da tabela ordenados por ordem crescente segundo o campo de nome.

```
SELECT tbl_alunos.*, tbl_contas.resumo from

tbl_alunos INNER JOIN tbl_contas ON tbl_contas.numContribuinte =
tbl_alunos.numContribuinte
```

Esta consulta une duas tabelas: a tabela alunos e outra tabela hipotética “contas”. Devolve todos os campos da tabela “alunos” e o campo de resumo da tabela “contas” vinculadas pelo “numContribuinte”, ou seja, irá extrair o resumo da conta de cada utilizador juntamente com os seus dados.

- INSERT: esta consulta é utilizada para inserir dados numa tabela de base de dados e a sua sintaxe é: INSERT INTO [tabela] (campos separados por vírgulas] VALUES [valores separados por vírgulas na mesma ordem dos campos]. Segue-se o exemplo com a tabela “tbl_alunos”:

```
INSERT INTO tbl_alunos (nome, apelidos, morada, numContribuinte,
data) VALUES ('Carlos', 'Santos Teste', 'base de dados local',
'999999999', '2021-02-02');
```

- UPDATE: esta função permite modificar um ou mais registos da base de dados e a sua sintaxe é: UPDATE [tabela] SET [campo] = [valor], [campo] = [valor], [campo] = [valor], [...] WHERE [opcional: condições].

  Exemplos:

```
UPDATE tbl_alunos SET apelidos = 'Pereira Teste'
```

  Esta consulta irá modificar todos os registos da tabela, ou seja, irá colocar o mesmo apelido ("Pereira Teste") a todos os registos.

```
UPDATE tbl_alunos SET numContribuinte = '999999999' , morada =
'Ruade Dona Estefânia 84' WHERE IDaluno = 1;
```

  Esta consulta atualiza o numContribuinte e a morada do aluno cujo ID seja 1.

  É muito importante ter cuidado ao enviar estas instruções, pois pode acontecer atualizar-se todos os registos da tabela sem se perceber.

- DELETE: esta consulta ajuda a apagar os registos de uma tabela e a sua sintaxe é: DELETE FROM [tabela] WHERE [opcional: condição]. Seguem-se dois exemplos:

```
DELETE FROM tbl_alunos
```

  Esta consulta vai excluir todos os registos da tabela.

```
DELETE FROM tbl_alunos WHERE nome = 'Carlos'
```

  Esta outra consulta apagará apenas os registos dos alunos com o nome "Carlos".

  Estas são as consultas mais utilizadas em SQL. Se a linguagem não for dominada.

  Um exemplo da utilização do mysqli_query:

```
<?php
$rs = mysqli_query($conn,'SELECT * FROM tbl_alunos');
```

```
?>
```

- **mysqli_num_rows**: devolve o número de registos devolvidos numa consulta (como o SELECT). Para isso é passado o resource que a consulta devolveu.

```
<?php
$rs = mysqli_query($conn,'SELECT * FROM tbl_alunos');
$registos = mysqli_num_rows($rs);
?>
```

- **mysqli_fetch_array**: recupera o registo do resultado da consulta (SELECT) como um array de chave → valor ou array numérico (posição → valor), ou devolve false, se não houver mais linhas. Move o ponteiro dentro do resultado para a frente, ou seja, avança para o registo seguinte.

  Para escolher que tipo de array é devolvido basta passar um segundo parâmetro na função:

  - MYSQL_ASSOC: devolve como chave de array → valor.
  - MYSQL_NUM: devolve o array com posição → valor.
  - MYSQL_BOTH: (valor padrão) devolve ambos.

```
<?php
$rs = mysqli_query($conn,'SELECT * FROM tbl_alunos');
while ($registo = mysqli_fetch_array($rs)) {
echo $registo[‘nome’].’<br/>’;
}
?>
```

  Este exemplo lê todos os registos devolvidos no resultado da consulta.

- **mysqli_fetch_assoc**: é igual à função anterior, com a exceção de que só pode devolver o array no formato de chave → valor. Move o ponteiro dentro do resultado para a frente, ou seja, avança para o registo seguinte.

- **mysqli_fetch_row**: tal como o mysqli_fetch_array, mas só devolve o array em formato numérico, ou seja, posição → valor. E move o ponteiro dentro do resultado para a frente, ou seja, avança para o registo seguinte.
- **mysqli_insert_id**: devolve o id criado no último insert da tabela.

```
<?php

mysqli_query($conn, "INSERT INTO tbl_alunos (nome, apelidos,
morada, numContribuinte, data) VALUES ('Carlos', 'Santos Teste',
'base de dados local', '999999999', '2021-02-02')");

echo "Último ID ".mysqli_insert_id($conn);

?>
```

- **mysqli_affected_rows**: devolve o número de registos afetados pela última consulta (INSERT, UPDATE ou DELETE).

```
<?php

mysqli_query($conn,”DETELE FROM tbl_alunos”);

echo ‘Foi removido ‘.mysqli_affected_rows ($conn) . ‘registos’;

?>
```

- **mysqli_real_escape_string**: escapa aos caracteres especiais de uma string a ser utilizada na instrução SQL, ou seja, muda os caracteres especiais utilizados no SQL (como as plicas enquanto carácter de escape) e é utilizado ao mesmo tempo que a função sprintf já abordada, que permite escrever a string e substituir os valores.

```
<?php

$query = sprintf("SELECT * FROM tbl_alunos WHERE nome=’%s’ or
apellidos = ‘%s’",

mysqli_real_escape_string($conn,$nome),

mysqli_real_escape_string($conn,$apelidos));

?>
```

Esta função é muito utilizada e muito útil, uma vez que impede de fazer um ataque por injeção SQL, que é um ataque informático às bases de dados para modificar, ler ou apagar ficheiros.

Com as funções estudadas, está preparado para criar as páginas em PHP com conexão e trabalho direto em MySQL.

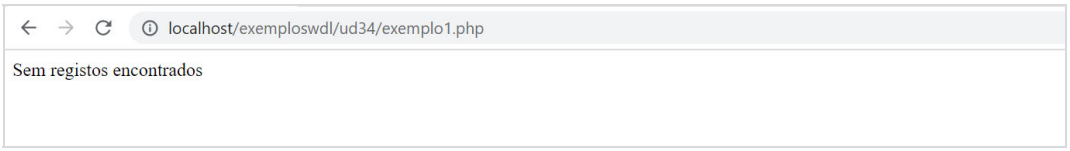
## Exemplo 01

Neste exemplo, serão listados todos os registos da tabela "tbl_alunos", utilizando tudo o que foi visto nesta unidade didática.

```
<html>
<body>
<p>
<?php
// utilizar PHP
$hostname = 'localhost';
$username = 'root';
$password = '[a senha definida na instalação]';
$dbname = 'escola';
$conn = @mysqli_connect($hostname, $username, $password);
if($conn) {
// selecionar a base de dados
if(mysqli_select_db($conn, $dbname) === TRUE) {
$sql = "SELECT * FROM tbl_alunos";
$result = mysqli_query($conn, $sql);
if($result) {
// se houver registos
if(mysqli_num_rows($result) !== 0) {
while($row=mysqli_fetch_array($result)) {
echo $row['numContribuinte'].' - '.$row['nome'].'
'.$row['apelidos'].'<br />';
}
} else {
echo 'Sem registos encontrados';
}
```

```
}
} else {
echo 'Falha ao selecionar a base de dados';
}
// fechar a conexão
mysqli_close($conn);
} else {
// falha na conexão
echo 'Falha na conexão';
}
?>
</p>
</body>
</html>
```

Neste caso, e como a tabela não tem registos, ao carregar a página, a visualização será:




## Exemplo 02

No exemplo seguinte irá preencher-se os dados da tabela "alunos", para os mostrar, e depois modificar e apagar alguns registos para os voltar a mostrar.

```
<html>
<body>
<p>
<?php
```

```
// utilizar PHP
$hostname = 'localhost';
$username = 'root';
$password = '[a senha definida na instalação]';
$dbname = 'escola';
$conn = @mysqli_connect($hostname, $username, $password);
if($conn) {
// selecionar a base de dados
if(mysqli_select_db($conn, $dbname) === TRUE) {
// inserir registos
$sql = "INSERT INTO tbl_alunos(numContribuinte, nome, apelidos)
VALUES ('999999999','Aluno1','Apelidos1'),
('988888888','Aluno2','Apelidos2'),
('977777777','Aluno3','Apelidos3')
";
if (mysqli_query($conn, $sql)){
echo 'Registos inseridos com sucesso!<br/><br/>';
}
$sql = "SELECT * FROM tbl_alumnos";
$result = mysqli_query($conn, $sql);
if($result) {
// se houver registos
if(mysqli_num_rows($result) !== 0) {
?>
<strong>Mostrar os dados após inserir os registos:</strong>
<table cellpadding="4" cellspacing="1" bor-der="1">
<tr>
<th>ID</th>
<th>Número de Contribuinte</th>
<th>Nome</th>
<th>Apelidos</th>
</tr>
```

```
<?php
while($row=mysqli_fetch_array($result)) {
?>
<tr>
<td><?php echo $row['IDaluno'];?></td>
<td><?php echo $row['numContribuinte'];?></td>
<th><?php echo $row['nome'];?></td>
<td><?php echo $row['apelidos'];?></td>
</tr>
<?php
}
?>
</table>
<?php
}
}
// mudar o registo com ID igual a 1
$sql = "UPDATE tbl_alunos SET nome = 'Nome alterado'
WHERE idaluno = 1";
mysqli_query($conn, $sql);
echo 'Eu atualizei o registo 1<br />';
// eliminar o registo 2
$sql = "DELETE FROM tbl_alunos WHERE IDaluno = 2";
mysqli_query($conn, $sql);
echo 'Eu apaguei o registo 2<br />';
$sql = "SELECT * FROM tbl_alunos";
$result = mysqli_query($conn, $sql);
if($result) {
// se houver registos
if(mysqli_num_rows($result) !== 0) {
?>
<strong>Mostrar novamente após as mudanças feitas:</strong>
```

```
<table cellpadding="4" cellspacing="1" bor-der="1">
<tr>
<th>ID</th>
<th>Número de Contribuinte</th>
<th>Nome</th>
<th>Apelidos</th>
</tr>
<?php
while($row=mysqli_fetch_array($result)) {
?>
<tr>
<td><?php echo $row['IDaluno'];?></td>
<td><?php echo $row['numContribuinte'];?></td>
<th><?php echo $row['nome'];?></td>
<td><?php echo $row['apelidos'];?></td>
</tr>
<?php
}
?>
</table>
<?php
} else {
echo 'Falha ao recolher o registo<br />';
}
} else {
echo 'Erro: A ação não pôde ser realizada<br />';
}
} else {
echo 'A atualização do registo falhou<br />';
}
mysqli_close($conn);
} else {
```

```
echo 'Falha ao conectar à base de dados<br />';
}
?>
</body>
</html>
```

# CONCLUSÃO

Com este sistema de conexão entre base de dados e PHP, aprendeu a criar páginas web totalmente dinâmicas, já que agora é capaz de ler e escrever tabelas.

Os sistemas internos ou CRM das empresas baseiam a sua estrutura num núcleo de base de dados onde gerem todos os dados dos funcionários, clientes, de faturação, entre outros.

# AUTOAVALIAÇÃO

1. **O que é o MySQL?**

   **a)** Um sistema de gestão de base de dados.

   **b)** Uma linguagem do script.

   **c)** Um sistema de entrega de correio.

   **d)** Um sistema de leitura de base de dados.

2. **Qual das seguintes afirmações não é válida?**

   **a)** O MySQL facilita a pesquisa em tabelas.

   **b)** O MySQL reduz o tempo de pesquisa de log, em comparação com XML.

   **c)** O MySQL é estruturado em tabelas.

   **d)** O MySQL não consegue criar um sistema de índice.

3. **O que é o campo collation no php MyAdmin?**

   **a)** Uma opção dos campos para comparar o valor com outro.

   **b)** Uma opção dos campos para indicar um valor padrão.

   **c)** Uma opção dos campos para indicar o tipo de codificação.

   **d)** Uma opção dos campos para indicar os registos.

**4. Qual é o nome do driver utilizado em PHP para aceder às bases de dados que não têm um driver específico?**

**a)** dBase.

**b)** coonectDB.

**c)** ODBC.

**d)** OCI8.

**5. Qual é a função da biblioteca de acesso PHP MySQL para conectar à base de dados?**

**a)** mysqli_conn.

**b)** connect_mysqli.

**c)** mysqli_connect.

**d)** connect_mysql.

**6. O que devolve a função mysqli_con_error?**

**a)** O erro criado pela conexão à base de dados.

**b)** O primeiro erro criado na página pelo MySQL.

**c)** Um evento de erro para exibir uma mensagem.

**d)** Nada, essa função não existe.

**7. Qual é a função utilizada entre PHP e MySQL para selecionar a base de dados dentro do MySQL?**

**a)** mysqli_select_bd.

**b)** mysqli_select_db.

**c)** mysqli_select.

**d)** select_mysql_bd.

8. **Quais são os dois parâmetros que devem ser passados para a função ao selecionar a base de dados na conexão MySQL?**

   **a)** O identificador da conexão e o nome da base de dados.

   **b)** O identificador de conexão e o ID da base de dados.

   **c)** Apenas o identificador de conexão.

   **d)** Apenas o identificador da base de dados.

9. **Qual é a função que fecha a conexão com a base de dados?**

   **a)** mysqli_close.

   **b)** mysqli_unset.

   **c)** close_mysqli.

   **d)** [identificador]->close();.

10. **O que devolve a instrução SQL SELECT?**

    **a)** Sempre true.

    **b)** Sempre false.

    **c)** Não devolve nada.

    **d)** Devolve um resource.

# SOLUÇÕES

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **1.** | a | **2.** | d | **3.** | c | **4.** | c | **5.** | c |
| **6.** | d | **7.** | b | **8.** | a | **9.** | a | **10.** | d |

# PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

Para descobrir todas as bibliotecas de conexão direta de base de dados, visite a seguinte página web:

- http://www.php.net/manual/pt_BR/refs.database.vendors.php

Para saber como usar o driver de conexão ODBC genérico, visite a seguinte página web:

- http://www.php.net/manual/pt_BR/ref.uodbc.php

# BIBLIOGRAFIA


- Cabezas, L. M. (2010), *Php 5.* Madrid: Anaya Multimedia.
- The PHP Group (2001), "SimpleXML". Disponível em: https://www.php.net/manual/pt_BR/book.simplexml.php Consultado a 23 de março de 2021.