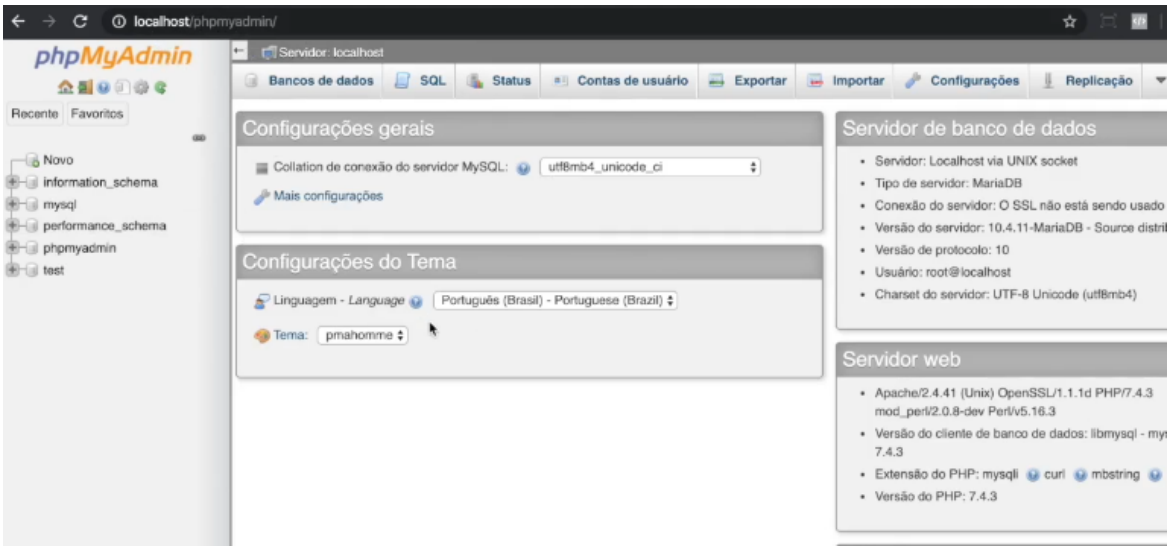
Introdução à Banco de Dados

Um banco de dados é o local onde ficam armazenados os dados que vamos usar no nosso sistema. É um serviço separado, ou seja, nos temos o Apache que vai rodar para interpretar as informações que estão na página, junto com o php que é o interpretador. E roda na porta 80 a parte disso temos um software que é o banco de dados, que guarda qualquer tipo de informação que vamos precisar para consultar no nosso sistema. Por exemplo, um sistema de controle de estoque, precisamos de algum lugar para armazenar esse estoque, então usamos o banco de dados. Como armazenamos essas informações? Com que estrutura? Que ordem? Ai você que vai desenhar seu banco de dados, você vai definir como quer armazenar essas informações. Existem atualmente dois principais tipos de banco de dados que é o que chamamos banco de dados relacional e banco de dados não relacional. Como o PHP, 99% dos casos trabalhamos com o banco de dados relacionais. Um dos bancos de dados mais utilizados com PHP e o MySQL. Tem também o MariaDB que é na verdade um melhoramento, do MySQL, com uso da mesma arquitetura, mesma estrutura, ou seja, mesmos comandos, etc.

Visão geral do PHPMyAdmin

Agora vamos ver um visão geral do software mais utilizado na manipulação de banco de dados que é o PHPMyAdmin. Normalmente se você instalou um xampp ou wamp ele vem junto, e ai para acessar basta com o servidor funcionando digitar localhost/phpmyadmin/ que vai levar diretamente para o sistema.



Não necessariamente você precisa instalar para conseguir manipular coisas com o banco de dados. Existem vários outros softwares que fazem absolutamente a mesma coisa, por exemplo, o MySQL Workbench que é um software que você instala no seu computador, muito similar o phpmyadmin. Temos uma lista de software como:

HeidiSQL - windows / Linux

SequelPro - Mac

PHPMyAdmin - online, funciona na verdade no seu próprio servidor interno, não precisa de internet para ele funcionar. Você só precisa instalar no seu computador o próprio servidor.

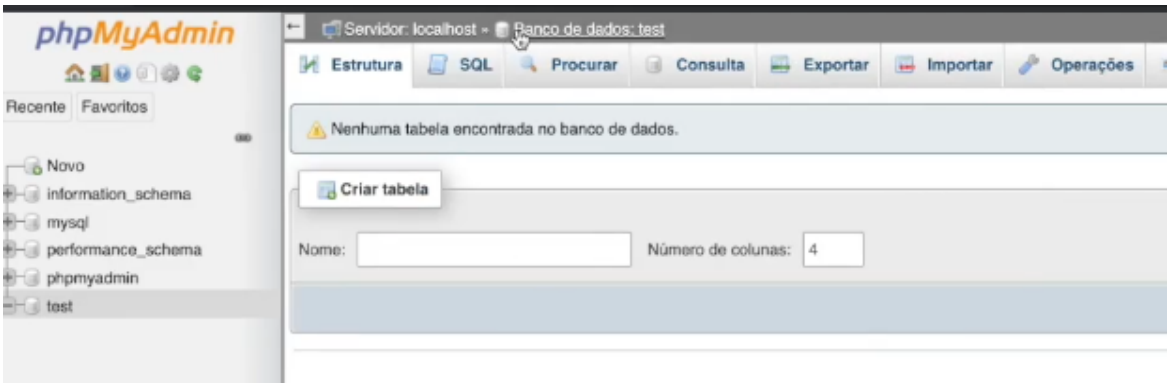
Vamos ver algumas regra básicas do phpmyadmin e como funciona o banco de dados internamente, antes de partirmos para o código.

Para conectar ao banco de dados devemos manter certo nível de privacidade, que precisa ser mantido. Para isso precisamos usar um usuário e uma senha. Então percebam se irmos na tela inicial temos a direita essa área:

Onde está **Usuário: root@localhost** na imagem acima, a palavra root é o usuário padrão que vem quando instalamos o servidor e a senha geralmente vem vazia. Mas quando tem pode ser usuário root e senha root também. Onde tem localhost significa o IP de onde está o rodando o servidor.

A esquerda da tela principal temos os banco de dados já carregados:

Ele cria um chamado test. Se clicarmos nele vai abrir o banco de dados test, conforme podem ver na parte superior Servidor: localhost Banco de dados: test, ilustrado na figura a seguir:

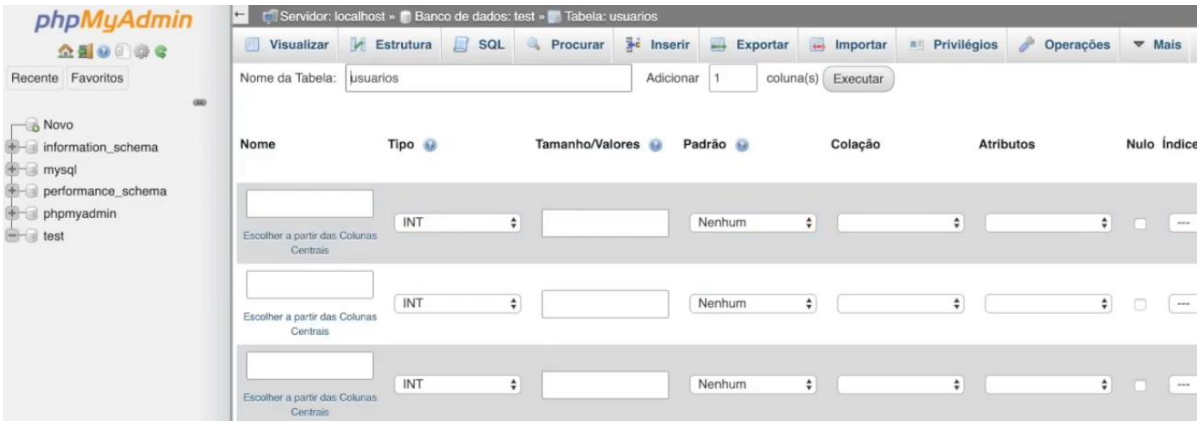


Observem que tem algumas abas e entre elas as duas principais está estrutura e procurar.

Na aba estrutura vai aparecer as tabelas no banco de dados. Como podem ver no banco de dados test não existe nenhuma tabela como está mostrando acima. Então num banco de dado nós temos varias tabelas, como no Excel, no banco de dados o processo é bem similar. Por exemplo, uma tabela vai armazenar os usuários do sistema, outra os produtos a venda na loja virtual, outra responsável pelo cadastro das vendas e assim por diante. Do mesmo jeito de uma tabela no Excel, ela tem que ter colunas e linhas. As colunas são o tipo de dados que vai armazenar e as linhas são os próprios dados.

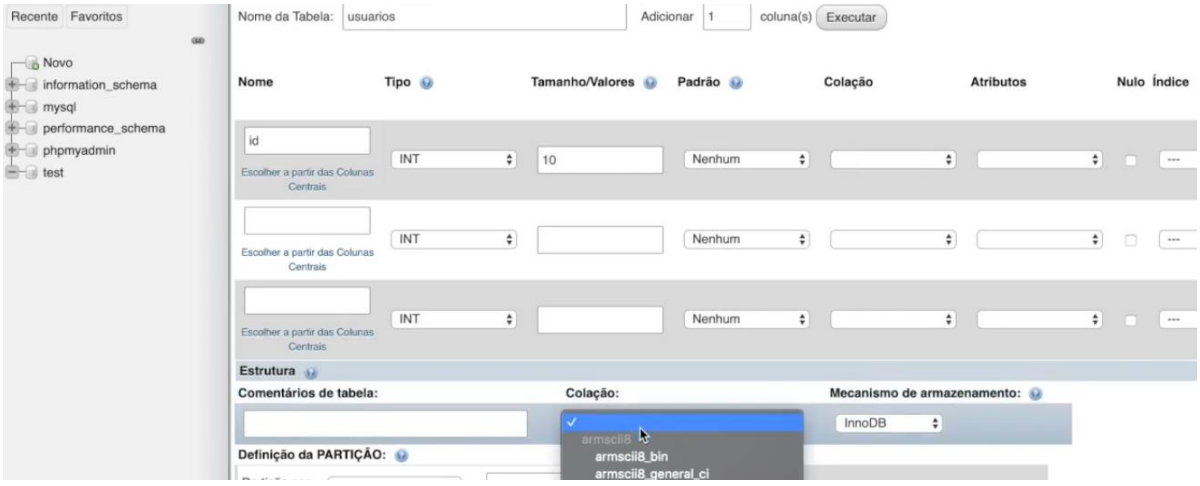
Então agora, vamos criar no banco de dados **tes**t uma tabela chamada **usuarios**. Normalmente o nome da tabela digitamos em minúsculo, sem caracteres especiais, sem acentos e também no plural. Por que no plural ? Por que vai armazenar vários usuários conforme nosso exemplo. E definimos inicialmente o número de colunas que a tabela vai ter, podendo mudar a hora que necessário esse número. Então vamos adicionar 3 colunas na tabela usuarios e confirmamos clicando no botão Executar, conforme mostra a figura a seguir:

Observem a próxima janela que aparece após confirmar no botão Executar:



Agora vamos criar o nome de cada coluna. Primeira coluna **id**, que geralmente colocamos em todas as tabelas que criamos. Por que id? Por que vai receber a identificação daquele registro específico. Por exemplo, nome Sandra, não vamos identificar um usuário pelo nome dele, por que podemos ter outras pessoa com o mesmo nome. Então identificamos um item pelo id, gera um número aleatório e fica sendo a representação, tipo CPF do usuário.

Observem que toda coluna tem um nome, um tipo (que dados serão armazenados: VARCHAR que significa texto - para frase, nome, e-mail. TEXT quando se tem um texto maior, mais de um parágrafo, múltiplas linhas. DATE quando se tem uma data específica que queremos salvar. DATETIME para armazenar a data e hora e minutos, segundos. FLOAT para números fracionários, INT para valores inteiro e assim por diante). Assim para o **id** vamos colocar tipo INT (inteiro) e colocamos um Tamanho/Valores que é o tamanho em caracteres que ele vai permitir, então se colocar o valor 3 vai permitir números que tenham no máximo 3 caracteres (999 por exemplo). No Padrão, normalmente não vai precisar mudar (deixa como Nenhum). Colação, é o tipo de armazenamento, tipo de incript que vai usar na coluna específica. Normalmente se coloca abaixo ao final da tabela como mostra a figura a seguir:

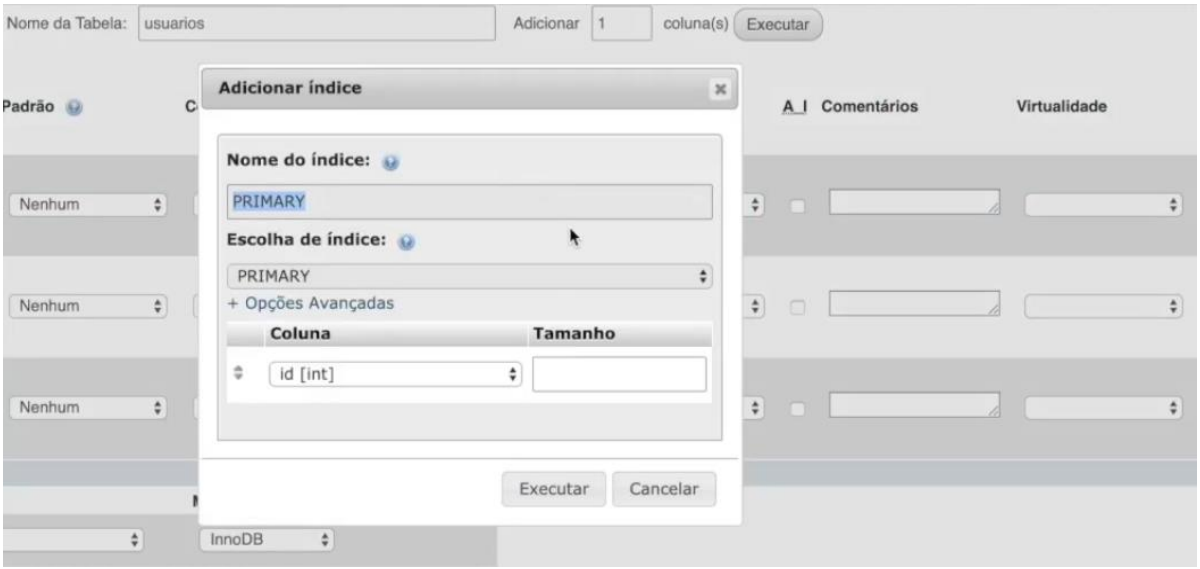


Atributos quase não usamos.

Campo Nulo, quando marcamos conforme mostra figura a seguir:

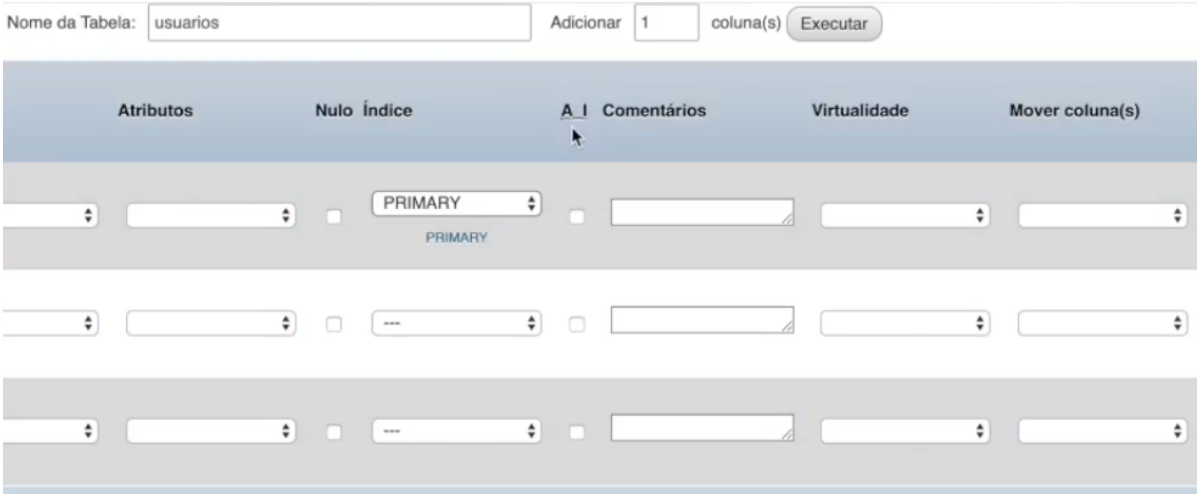
Se inserirmos um registro e não especificar o id dele, vai vir como nulo, não vai vir com valor algum. Então em alguns campos vamos querer isso já no id não é o que se quer, não selecionamos essa opção, pois todos os usuários tem que ter uma identificação. Na opção Índice vamos usar muito, principalmente no id, observem a figura que ilustra:

Para o campo id, vamos colocar como índice primário. Isso quer dizer que vamos usar como o campo id como campo de referencia de identificação primária para o registro que vamos usar.



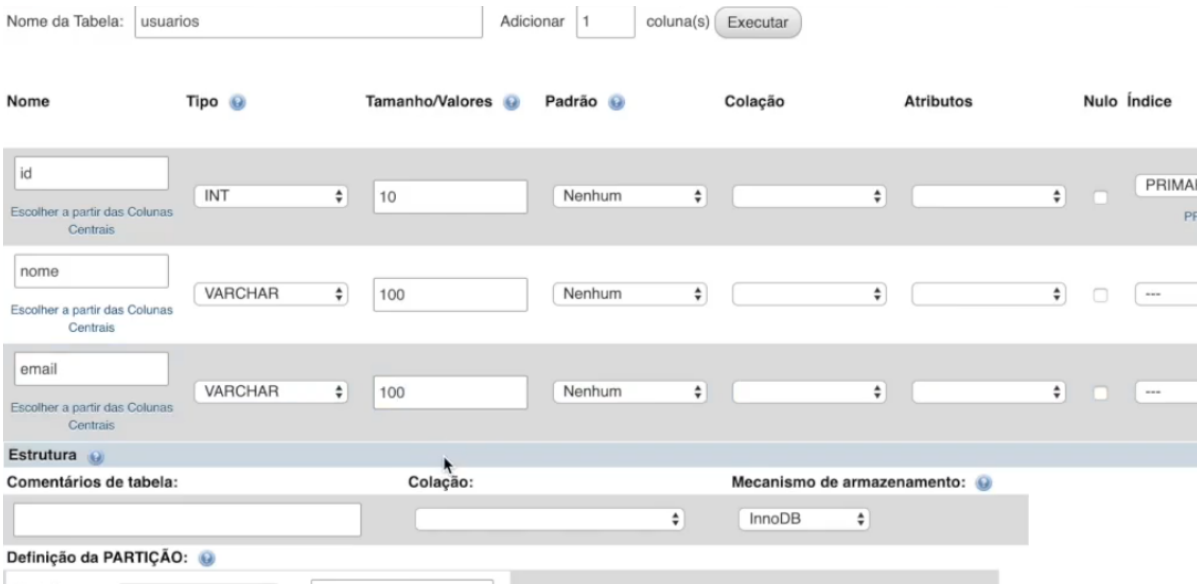
Após escolher a opção primary, confirmamos no botão Executar conforme figura acima.

A próxima opção é o A.I que significa auto increment (incremento automático):

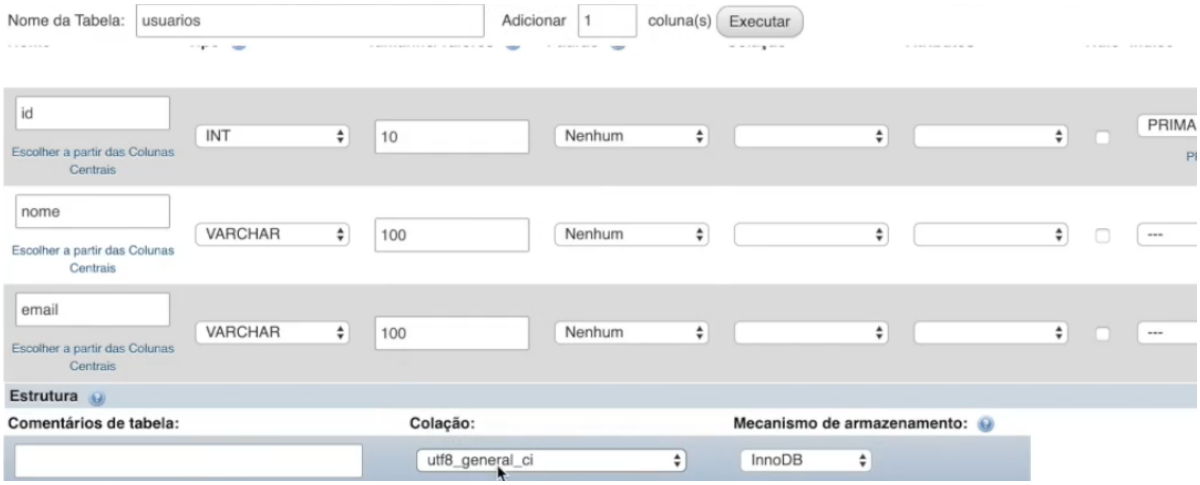


Ou seja, é um número que vai incrementar automaticamente. Isso significa se selecionar essa opção, quando você adicionar, por exemplo, o primeiro usuário vai estar lá usuário com id igual a 1. Quando adicionar o segundo usuário vai para o número 2 e assim sucessivamente. Os demais campos a seguir não são muito importantes então deixamos para mais a diante ver.

Observem o preenchimento dos outros dois campos da tabela na figura a seguir:



Agora observem ao final na opção **Colação**:



A recomendação geral sempre em termos de encriptação de qualquer coisa é a opção **utf_general_ci** ou então pode também utilizar a opção **utf8mb4_general_ci.**

O que se declara no MySQL como utf8 na verdade não é UTF-8, é apenas um pedaço dele. Para consertar este erro, a partir da versão 5.5, o MySQL implementou o padrão completo indo de 1 até 4 bytes e como já havia usado o nome utf8 chamou sua nova implementação de utf8mb4. Resumindo o utf8 do MySQL não é UTF-8 e o utf8mb4 segue totalmente o padrão UTF-8."

Inclusive no site PHP The Right Way (PHP do Jeito certo), recomenda-se que se use esse padrão do codificação (utf8mb4).

INFO:

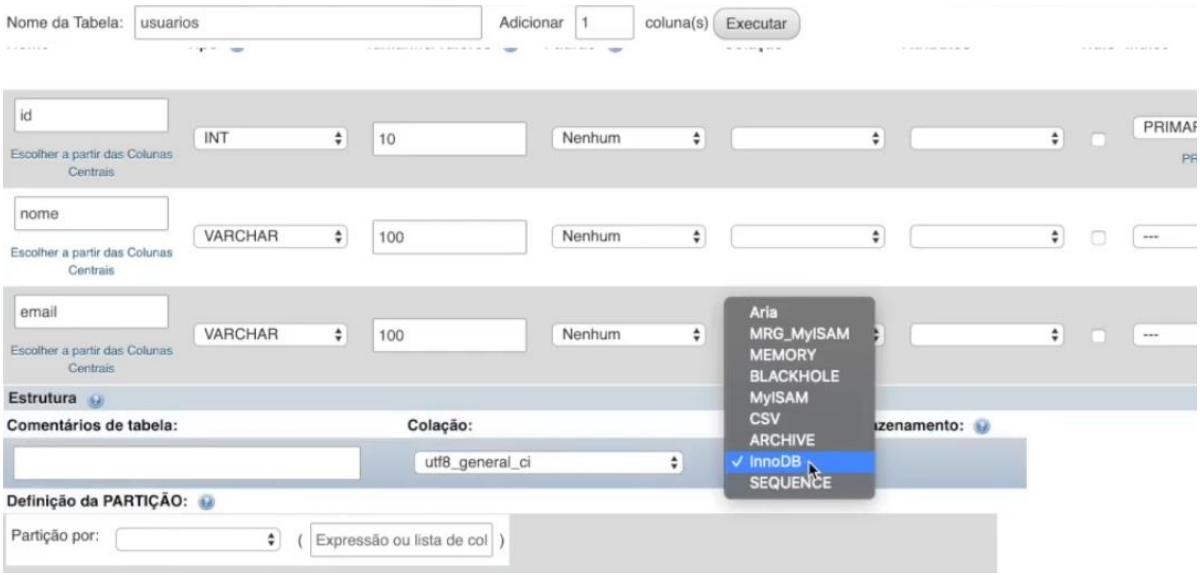
https://phptherightway.com/

Mais info:

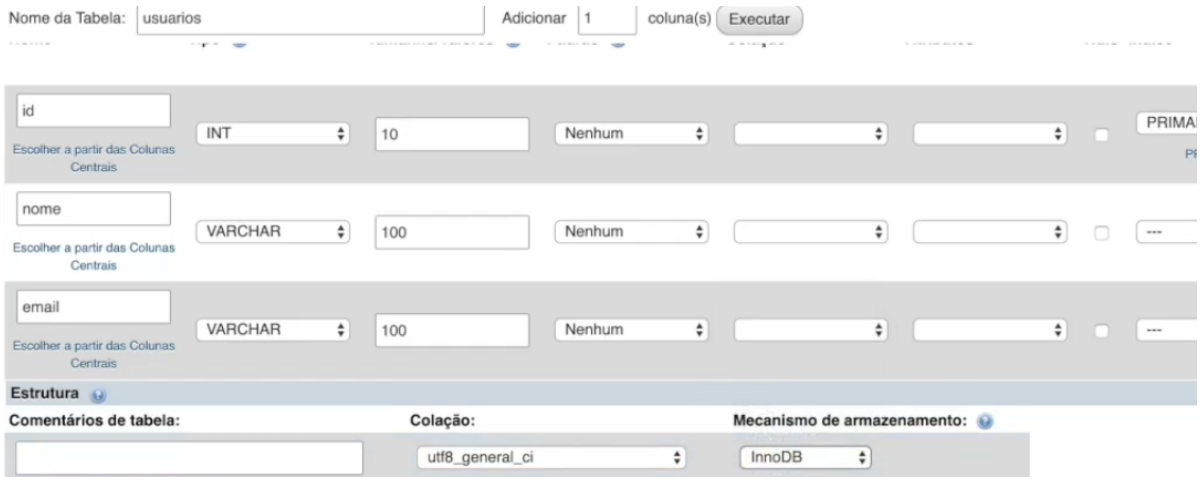
https://medium.com/@adamhooper/in-mysql-never-use-utf8-use-utf8mb4-11761243e434

https://pt.stackoverflow.com/questions/124831/quais-as-diferen%C3%A7as-entre-utf8-e-utf8mb4

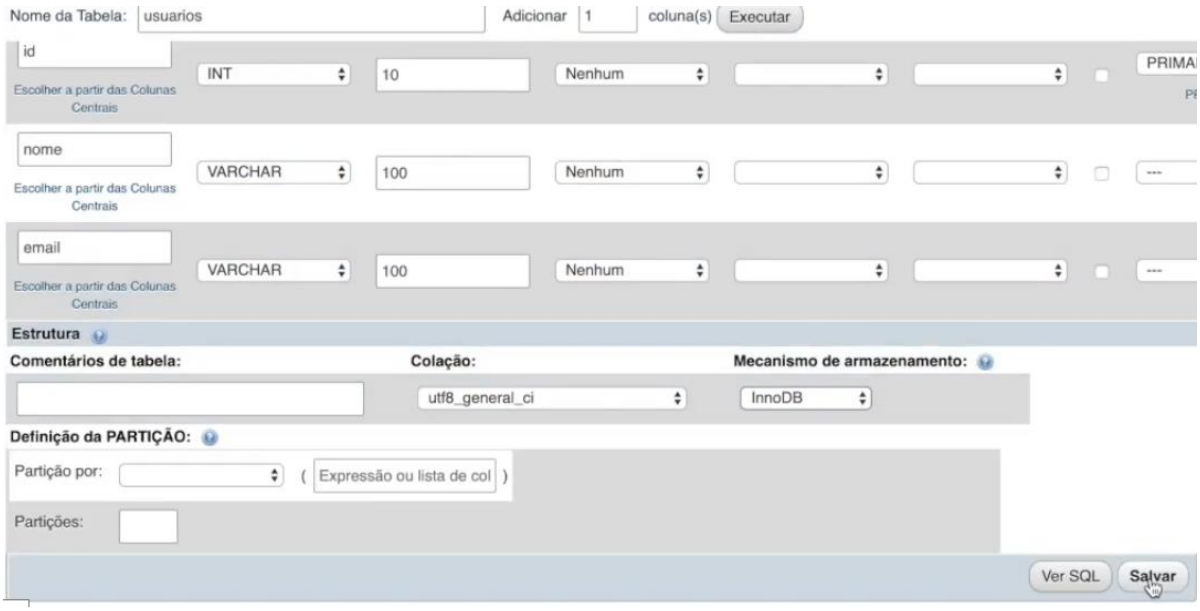
No Mecanismo de armazenamento temos o InnoDB e MyISAM.



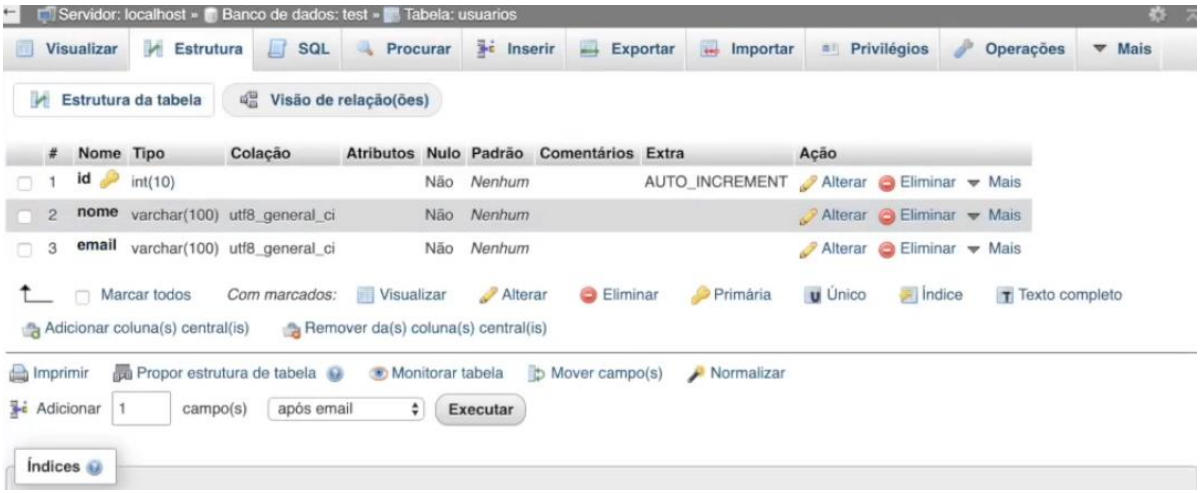
Esse dois mais recomendados. Mas o mais utilizado é o InnoDB:



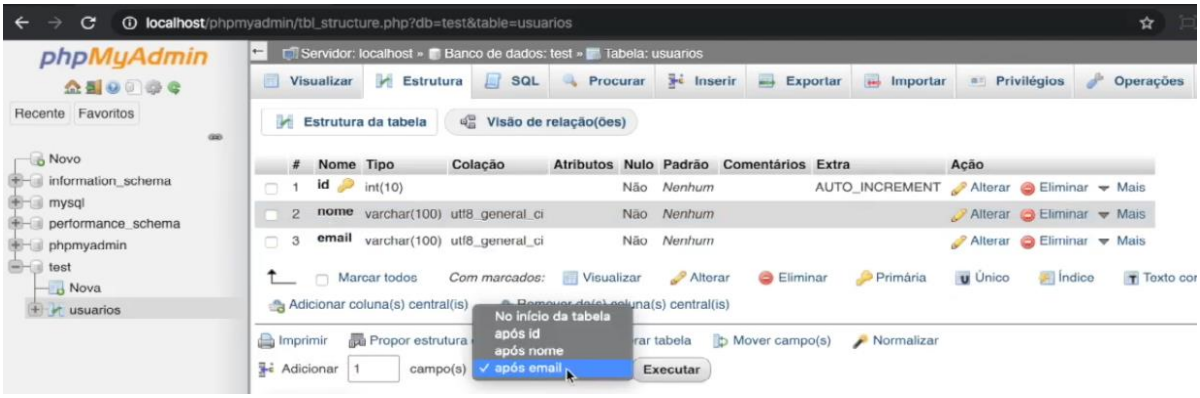
Feito isso acabamos de montar a estrutura da tabela usuários, agora é só clicar o botão Salvar:



Observem a chave amarela ao lado do campo id, que indica chave primária.

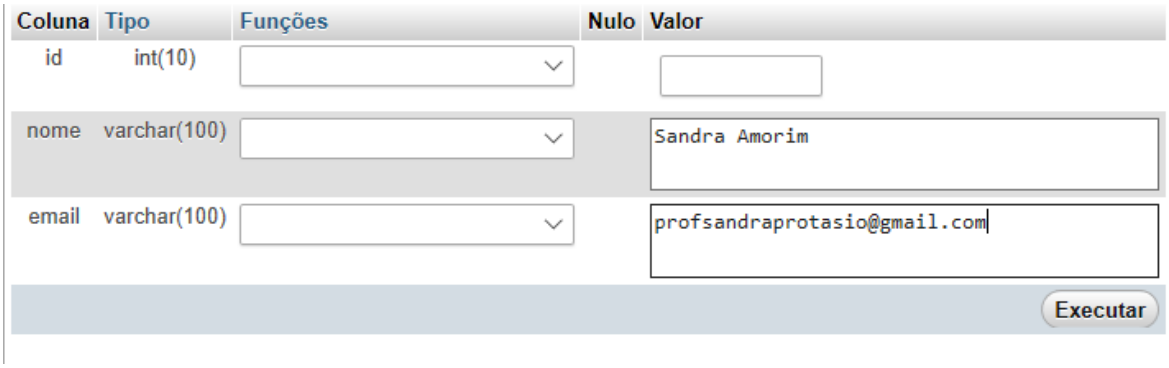


Como podem ver, podemos alterar, eliminar, adicionar mais colunas e assim por diante. Se quiser adicionar mais um campo após o campo email, por exemplo, observem na parte inferior da tabela:

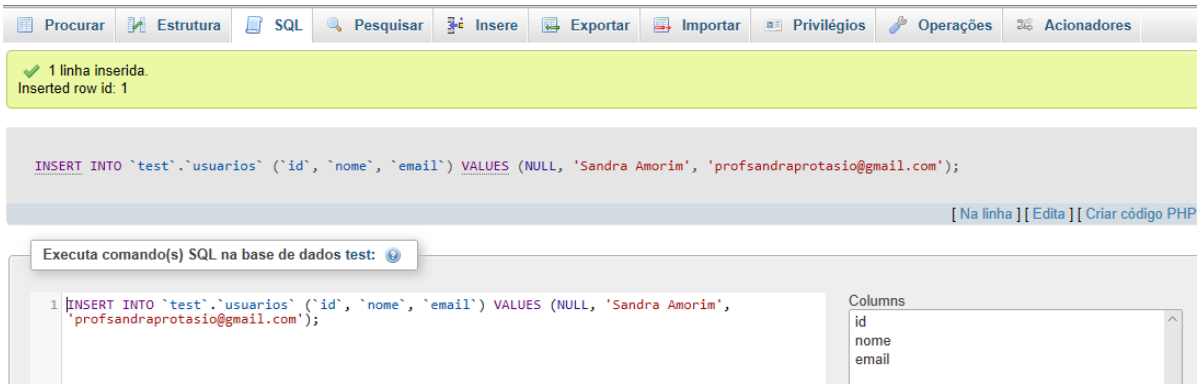


E após adicionar o número de campos para inserir e a opção após email é só clicar confirmando o botão Executar.

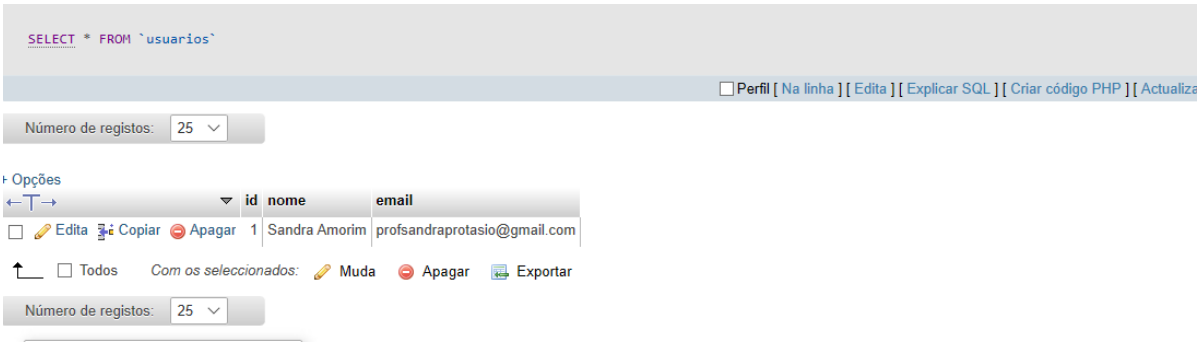
Agora como podemos inserir um dado nessa tabela? Basta ir à aba Inserir na parte superior abre a seguinte janela:



Observem que o id como é chave primária e auto incremento não especificamos, ele mesmo vai cuidar. Preenchemos os outros dois campos e clicamos em Executar.



Observe que mostra o comando que o SQL fez para criar a adição dos dados. Agora como podemos ver os dados inseridos? Basta irmos à coluna à esquerda, clicar no nome do banco de dados test e vai abrir duas opções, que é a aba visualizar ou estrutura.



Observe o comando SELECT * FROM 'usuarios', que mostra todos os dados de uma tabela.

Se quiserem incluir mais dados na tabela usuario, basta ir à aba Inserir ou SQL.

Conectando ao Banco de Dados com PDO

E agora vamos conectar o nosso banco de dados test com o nosso sistema. Para conectar o banco de dados com o PHP, existe varias forma de fazer isso, sendo a mais utilizada hoje em dia com o PDO. O PDO é uma biblioteca que normalmente já vem habilitada no PHP, é compatível com vários tipos de banco de dados e entre eles o próprio MySQL. O PDO é uma classe que vamos instanciar ela, ou seja, gerar um objeto.

$pdo = new PDO();

A variável $pdo, não precisa ser com esse nome, geralmente usa-se assim.

Passamos alguns itens no construtor dessa classe, que é uma série de comando que especificamos os dados iniciais de conexão do nosso banco de dados:

O primeiro parâmetro é o tipo de banco de dados(MySQL), seguido do dbname e o nome do nosso banco de dados, **host**, ou seja, onde esta funcionando esse banco de dados, no nosso caso no localhost ou o ip da máquina 127.0.0.1.

Depois precisamos dizer para ele o usuário e a senha:

No segundo parâmetro é o usuário, que por padrão na hora de instalação, vem como root, como já explicado no inicio dessa apostila.

Terceiro parâmetro e a senha, que vem vazia (""), em alguns caso pode ser root também essa senha, cabe fazer o teste na hora.

$pdo = new PDO("**mysql**:dbname=test; **host**=localhost", "root", "");

Quando não tem senha pode ser ocultado o parâmetro da senha:

$pdo = new PDO("**mysql**:dbname=test; **host**=localhost", "root");

Feito isso podemos testar essa conexão.

Arquivo **index.php**

<?php

$pdo = new PDO("**mysql**:dbname=test; **host**=localhost", "root", "");

?>

Se não mostrar nenhum erro está correto à conexão.

Vamos fazer uma consulta aos dados no nosso banco de dados. Vamos criar a variável $sql e montamos a SQL com o nosso $pdo seguido do método query() que é para fazer uma consulta ao banco de dados. Então, sempre que verem query() é um comando para inserir, consultar, alterar ou deletar.

Então vamos inserir um comando inicialmente para consultar no query():

$sql = $pdo->query('SELECT * FROM usuarios');

Depois dado o comando, vamos mostrar esses dados na tela:

$dados = $sql->fetchAll();

fetchAll() quer dizer: pegue todos os dados.

print_r($dados);

Observem como fica o código completo:

Arquivo **index.php:**

<?php

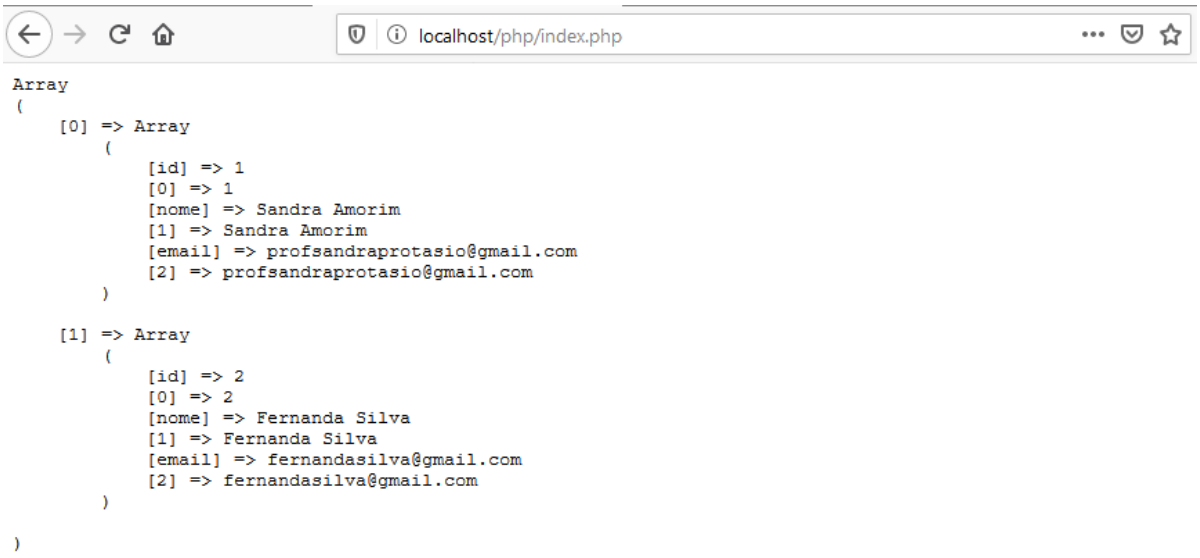
$pdo = new PDO("mysql:dbname=test;host=localhost", "root", "");

$sql = $pdo->query('SELECT * FROM usuarios');

$dados = $sql->fetchAll();

```
echo '<pre>';

print_r($dados);

?>
```

Executando nos retorna todos os registros da tabela usuarios, conforme mostra a figura abaixo:



Percebam que está repetindo, por exemplo, id => 1, [0] => 1 assim por diante. Isso aconteceu porque não definimos nenhum tipo de associação pra ser feita. Vamos ao fecchAll() para definir, e dentro vamos usar uma constante estática do PDO, ou seja, isso vai fazer que não mostre essa duplicidade:

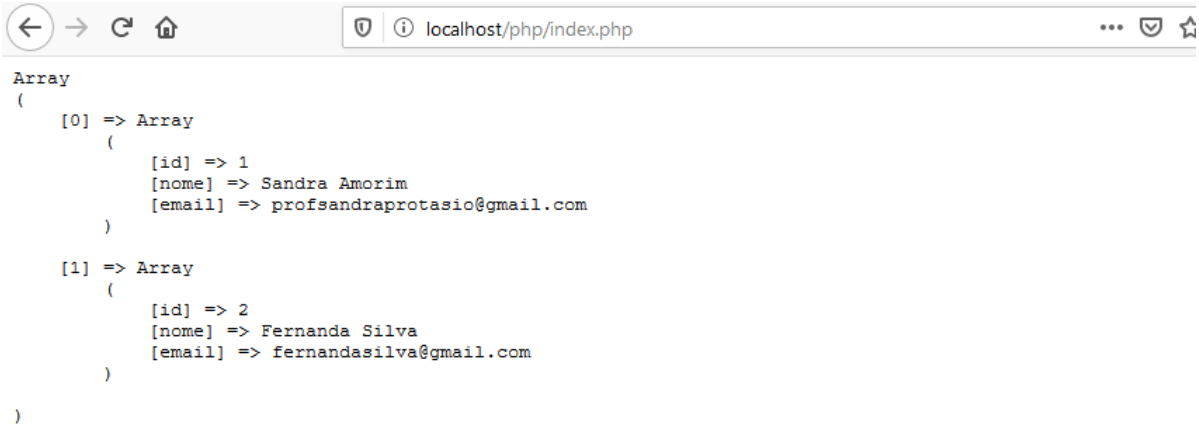
```
$dados = $sql->fetchAll( PDO::FETCH_ASSOC );
```

Observem como fica o código:

Arquivo **index.php**

```
<?php

$pdo = new PDO("mysql:dbname=test;host=localhost", "root", "");

$sql = $pdo->query('SELECT * FROM usuarios');

$dados = $sql->fetchAll( PDO::FETCH_ASSOC );

echo '<pre>';

print_r($dados);

?>
```

Atualizando a página percebam que não houve mais essa duplicidade dos dados:



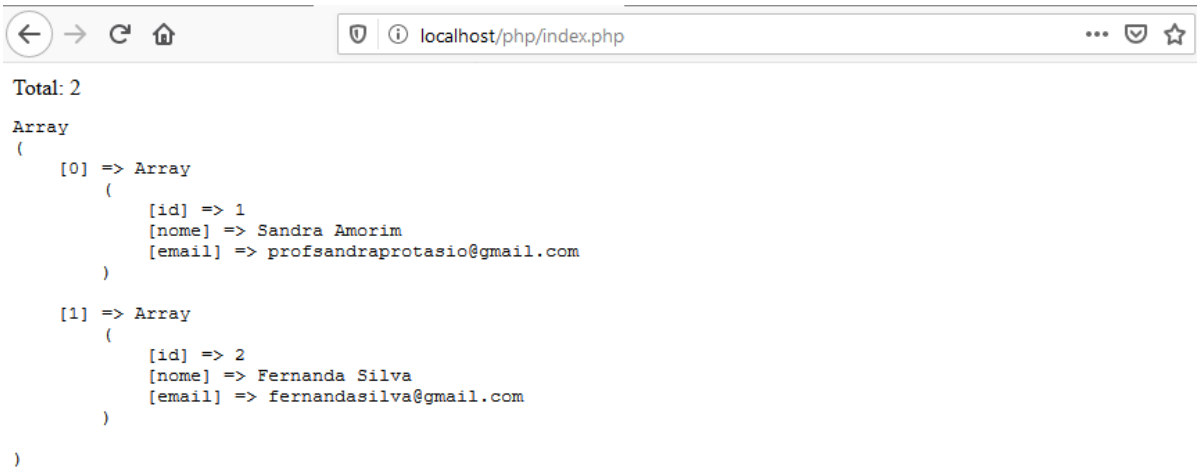
Através desse sistema php foi possível pegar os dados diretamente do nosso banco de dados test.

Se quisermos saber quantos registros temos na tabela usuários usamos o rowCount() onde row igual a linhas e count contagem, contagem de linhas.

Arquivo **index.php**

```
<?php

$pdo = new PDO("mysql:dbname=test;host=localhost", "root", "");

$sql = $pdo->query('SELECT * FROM usuarios');

$dados = $sql->fetchAll( PDO::FETCH_ASSOC );

echo "Total: ".$sql->rowCount();

echo '<pre>';

print_r($dados);

?>
```

Atualizando a página, observem o total de registros conforme a consulta na tabela usuarios:



Com isso fizemos o básico de uma consulta em php.

Conceito CRUD

CRUD é um dos conceitos mais importantes da programação como um todo.

C -> CREATE

R -> READ

U -> UPDATE

D -> DELETE

Esses 4 comandos são o que vai prevalecer nos programas que fizerem daqui por diante.

Por exemplo, um sistema de controle de estoque, nada mais é do que um CRUD, pois adiciona produtos, lê as informações dos produtos, altera e deleta produtos.

Um sistema de agenda de contatos vai ter um CRUD e assim por diante.

Esses conceitos são muito importantes, é a base dos sistemas que vocês venham a fazer.

Então vamos aplicar isso no nosso sistema, onde vamos inserir usuários novos, ler o usuários que estão cadastrados, alterar informações dos usuários e deletar informações desse registros.

Agora vamos separar do nosso arquivo index.php, os dados de configuração.

Para isso vamos criar um arquivo que vai se chamar **config.php** e dentro dele vamos colocar assim:

Arquivo **config.php**

<?php

```
//Posso só deixar conforme linha a seguir:
//$pdo = new PDO("mysql:dbname=test;host=localhost", "root", "");

//Ou, posso fazer assim também essa conexão com o banco de dados, vc escolhe qual achar melhor
$db_name = 'test';
$db_host = 'localhost';
$db_user = 'root';
$db_pass = '';
$pdo = new PDO("mysql:dbname=$db_name;host=$db_host", $db_user, $db_pass);
?>
```

No arquivo **index.php** fizemos assim:

```
<?php
require 'config.php';
?>
<html>
   <head>
      <meta charset="UTF-8">
   </head>
<a href="adicionar.php"> Adicionar Usuários </a>
<br/><br/>
<table border='1' width='100%'>
   <tr>
      <th>ID</th>
      <th>NOME</th>
      <th>EMAIL</th>
      <th>AÇÕES</th>
```

```
    </tr>
</table>
</html>
```

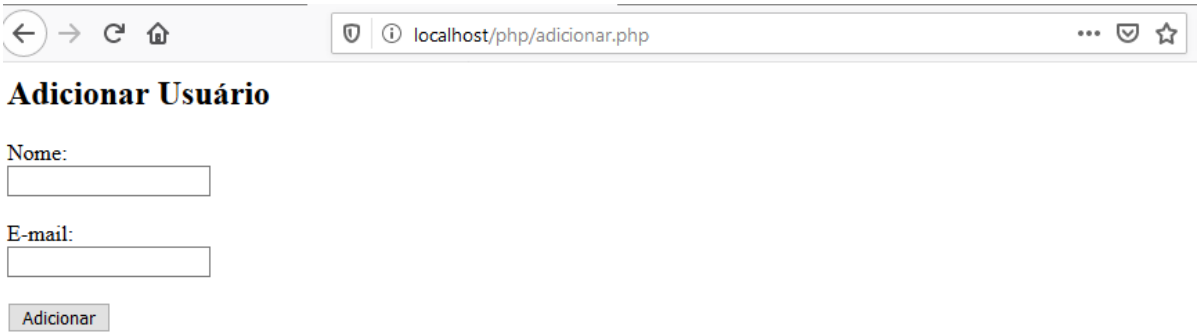
Create: Inserindo dados com PDO

Vamos focar no primeiro item do CRUD que é o create. Para isso vamos no criar o arquivo adicionar.php, que vai ter os campos para inserir os dados na tabela usuarios.

Arquivo **adicionar.php**

```
<html>
    <head>
        <meta charset="UTF-8">
    </head>
<h2> Adicionar Usuário </h2>
<form method="POST" action="adicionar_action.php">
    <label>
        Nome: <br/>
        <input type="text" name="name"/>
    </label><br/><br/>
    <label>
        E-mail:<br/>
        <input type="email" name="email"/>
    </label><br/><br/>
    <input type="submit" value="Adicionar"/>
</form>
```

Esse arquivo gerar a página igual a figura a seguir:



Quando clicar no botão Adicionar. vai enviar os dados dos 2 campos do formulário para o arquivo **adicionar_action.php.**

Vamos agora criar esse arquivo adicionar_action.php, para isso vamos lembrar que esse arquivo vai precisar acesso ao banco de dados test, então vamos precisar inserir nesse arquivo o require 'config.php'. Observem que no arquivo adicionar.php feito anteriormente não foi necessário utilizar a conexão (require 'config.php'), pois se trata apenas de um formulário.

Arquivo **adicionar_action.php**

```
<?php

//conexão com o banco de dados

require 'config.php';



$name = filter_input(INPUT_POST, 'name');

$email = filter_input(INPUT_POST, 'email',FILTER_VALIDATE_EMAIL);

if($name && $email){

    //query de inserção

    //$pdo->query("INSERT INTO usuarios (nome, email) VALUES ('$name' '$email') ");

    //A seguir a melhor forma de se fazer essa inserção, através do chamado PDOStatement:

    $sql = $pdo->prepare("INSERT INTO usuarios (nome, email) VALUE (:name, :email)");
//templates

    $sql->bindvalue(':name', $name);

    $sql->bindvalue(':email', $email);

    //Na linha a seguir vamos executar, ou seja mandar esses dados para tabela usuarios

    $sql->execute();
```

```
    //Direciono para voltar para página index.php
    header("Location: index.php");
    exit;

} else {
    header("Location: adicionar.php");
    exit;
}
?>
```

Feito esse arquivo, agora vamos na página index.php e clicamos no link, botão Adicionar Usuários, que direcionará para a página adicionar.php o que possibilitará adicionar novos usuários na tabela usuarios. Adicionamos um usuário, e observem que após a inserção dos dados, direciona para a página index.php, que por enquanto não fizemos para mostrar nessa página os dados armazenados na tabela usuarios. Para ver se adicionou mesmo o usuário cadastrado, vamos no phpMyAdmin e atualizamos a página e clicamos em visualizar para fazer essa verificação.

Observem a explicação de cada passo no código do arquivo **adicionar_action.php:**

Fazer a inserção dessa forma:

```
$pdo->query("INSERT INTO usuarios (name, email) VALUES ('$name' '$email') ")
```

deixa o sistema vulnerável a ataques. No lugar disso usamos o chamado **PDOStatement**. Onde vamos usar várias etapas, onde vamos montando aos poucos nossa query e depois executamos essa query. Começamos a montagem com:

```
$sql = $pdo->prepare();
```

Observem que não usei o método query() e sim o **prepare()**. E nesse método prepare()mandamos o esboço da query():

```
$sql = $pdo->prepare("INSERT INTO usuarios (nome, email) VALUE (:name, :email)");
```

O que significa no código acima :name, :email ? Significa que estamos montando um templates desses campos que depois vamos subtituir pelo valor real desses campos. A seguir fazemos essa subtituição por esse valores reais, que é a associação dos respectivos valores.

```
$sql->bindvalue();
```

O **bindvalue()** é um método do SQL que tem 2 parâmetros. O primeiro parâmetro quem quero modificar e o segundo parâmetro por quem quero trocar., ou seja:

$sql->bindvalue(':name', $name);

Além do bindvalue() temos outro o bindParam(). A diferença entre eles é simples, no bindvalue podemos estar associando o valor, veja exemplo:

$sql->bindvalue(':name', 'Sandra') que vai funcionar do mesmo jeito, por que está associando o valor 'Sandra' com :name.

E bindParam() apesar de parecer igual o funcionamento é um pouco diferente:

$sql->bindParam(':name', $name); nele estamos associando o próprio parâmetro, ou seja, a própria variável $name com o :name. Isso quer dizer se depois desse bindParam fizermos isso na sequência:

$name = 'Sandra Amorim'; vai trocar o valor da variável na query também, mesmo sendo depois da query essa alteração no valor.

O que é difirente no bindvalue é se fizermos assim:

$sql->bindvalue(':name', $name);

$name = 'Sandra';

Não vai alterar o valor recebido na query, ou seja, recebido na linha onde temos $sql->bindvalue(':name', $name); mesmo tendo a variável recebido outro valor logo depois.

Então na maioria dos casos vamos estar usando o bindvalue.

O que viram até aqui é para montar a query como PDOStatement, onde fizemos a substituição para tornar a query de forma segura de ataques. Agora em diante vamos para parte do código que executa essa query:

$sql->execute();

Depois que executou, fez a inserção dos dados temos que direcionar para a página index.php, é onde mostra os dados armazenados na tabela usuarios:

header("Location: index.php");

exit;

Agora vamos validar o e-mail cadastrado, por exemplo, não queremos que seja cadastrados 2 e-mail iguais na tabela usuario, como fizemos essa validação?

Vamos fazer no próprio sistema essa validação, para isso antes de adicionar verificamos se tem alguém já no sistema com o mesmo e-mail. Então, recebemos e fizemos essa verificação antes. Observem essa verificação no arquivo adicionar_action.php onde está digitado em vermelho:

<?php

```
//conexão com o banco de dados
require 'config.php';
$name = filter_input(INPUT_POST, 'name');
$email = filter_input(INPUT_POST, 'email',FILTER_VALIDATE_EMAIL);

if($name && $email){
    //validação do e-mail:
    $sql = $pdo->prepare("SELECT * FROM usuarios WHERE email = :email"); //sempre qdo vamos inviar dados, fazer com prepare()
    $sql->bindvalue(':email', $email);
    $sql->execute();

    if($sql-> rowCount() === 0){
    //query de inserção
    $sql = $pdo->prepare("INSERT INTO usuarios (nome, email) VALUE (:name, :email)"); //templates
    $sql->bindvalue(':name', $name);
    $sql->bindvalue(':email', $email);
    $sql->execute();
    header("Location: index.php");
    exit;

    } else {
        header("Location: adicionar.php");
        exit;
    }
} else {
    header("Location: adicionar.php");
```

```
    exit;
}
```

Read: Lendo dados com PDO

Vamos agora ler as informações que estão no banco de dados e exibir na página do index.php. Para isso vamos primeiro no código do index.php e digitar o código para buscar esses dados. Dentro da área do php no index.php digitamos:

Arquivo **index.php**

```
<?php
require 'config.php';
$lista = []; //array começa vazio
$sql = $pdo->query("SELECT * FROM usuarios");
if($sql->rowCount() > 0){
    $lista = $sql->fetchAll(PDO::FETCH_ASSOC);
}
?>

<html>
    <head>
        <meta charset="UTF-8">
    </head>

<a href="adicionar.php"> Adicionar Usuários </a>
</br></br>
<table border='1' width='100%'>
    <tr>
        <th>ID</th>
        <th>NOME</th>
```

```
        <th>EMAIL</th>
        <th>AÇÕES</th>
    </tr>
    <?php foreach($lista as $usuario): ?>
        <tr>
          <!-- <td> <?php echo $usuario['id']; ?> </td> -->
           <td> <?=$usuario['id'];?> </td>
           <td> <?=$usuario['nome'];?> </td>
           <td> <?=$usuario['email'];?> </td>
           <td></td>
        </tr>
    <?php endforeach; ?>
</table>
```

Agora observem que os registros agora aparecem na página index.php.

Agora vamos para um segundo passo que é fazer as ações, ou seja, os botões na última coluna a direita da tabela nesse arquivo. Cada um desses registros vai ter um botão para alterar e excluir. Para isso voltamos no arquivo e vamos colocar dentro dessa última coluna o código em html para visualizar esse botões com link para os próximos arquivos editar.php e excluir.php. Veja essa alteração no index.php onde está digitado em vermelho:

```
<?php
require 'config.php';
$lista = []; //array começa vazio
$sql = $pdo->query("SELECT * FROM usuarios");
if($sql->rowCount() > 0){
    $lista = $sql->fetchAll(PDO::FETCH_ASSOC);
}

?>
<html>
```

```
    <head>
        <meta charset="UTF-8">
    </head>


<a href="adicionar.php"> Adicionar Usuários </a>
</br></br>
<table border='1' width='100%'>
    <tr>
        <th>ID</th>
        <th>NOME</th>
        <th>EMAIL</th>
        <th>AÇÕES</th>
    </tr>
    <?php foreach($lista as $usuario): ?>
        <tr>
          <!-- <td> <?php echo $usuario['id']; ?> </td> -->
           <td> <?=$usuario['id'];?> </td>
           <td> <?=$usuario['nome'];?> </td>
           <td> <?=$usuario['email'];?> </td>
           <td>
               <a href="editar.php">[Editar]</a>
               <a href="excluir.php">[Excluir]</a>
           </td>
        </tr>
    <?php endforeach; ?>
</table>
```

Agora para editar devemos informar quem queremos editar. Por exemplo, tipo editar.php?id = 2. Então vamos fazer isso no link no arquivo index.php:

<a href="editar.php**?id=<?=$usuario['id'];?>**">[Editar]

Observem no link após o nome do arquivo, foi colocado **?id=** e após pegamos o script do registro do campo id, **<?=$usuario['id'];?>** acrescentamos, pois é ele que tem o id desse registro.

Do mesmo jeito vamos fazer para o link do excluir.php.

Observem na página fazendo um teste clicando nos botões, podem ver na url aparece ao final o id do respectivo usuário dessa linha, por exemplo, editar.php?id=2 na url.

Update: Atualizando dados com PDO

Agora vamos fazer o update do nosso CRUD, que é o ato de atualizar um dado específico. Esse update é feito em duas etapas, por que no adicionar precisamos consultar o banco de dados e mostrar nos campos formulário o nome e e-mail do registro consultado. Vamos criar o arquivo chamado editar.php e vai ser muito similar ao adicionar.php e por isso vamos copiar o código desse arquivo no editar.php e fazer as alterações necessárias. Para fazer o preenchimento dos campos no editar.php, com os dados, primeiro devemos verificar se recebemos o id:

Arquivo editar.php

```
?php
    require 'config.php';
    //1º passos
    $info = []; //Vai ter as informações dos usuários
    $id = filter_input(INPUT_GET,'id');
    if($id){
        $sql = $pdo->prepare("SELECT * FROM usuarios WHERE id = :id");
        $sql->bindvalue(':id', $id);
        $sql->execute();
        if($sql->rowCount() > 0){
            $info = $sql->fetch( PDO::FETCH_ASSOC ); //Vai pegar o primeiro item, aqui vai preencher a $info


        } else {
            header("Location: index.php");
```

```
            exit;
        }
    } else {
        header("Location: index.php");
        exit;
    }
?>
<html>
    <head>
        <meta charset="UTF-8">
    </head>

<h2> Editar Usuário </h2>

<form method="POST" action="editar_action.php">
    <label>
        Nome: <br/>
        <!-- 2º Passo adicionar um value... -->
        <input type="text" name="name" value="<?=$info['nome'];?>"/>
    </label><br/><br/>

    <label>
        E-mail:<br/>
        <input type="email" name="email" value="<?=$info['email'];?>"/>
    </label><br/><br/>
    <input type="submit" value="Editar"/>
</form>
```

Feito o arquivo editar.php, agora vamos na página index.php e clicamos no botão editar de alguns dos registro e fizemos a verificação se foi os dados conforme o id do usuário cadastrado para o formulário do arquivo editar.php. Feito isso, que foi o primeiro passo de editar, agora vamos para o segundo passo que é salvar essas alterações no banco de dados na tabela usuarios. Para isso vamos criar o arquivo editar_action.php. Antes, para receber esse processo de alteração nesse novo arquivo, ele tem que receber o id, além do nome e e-mail. Para isso precisamos ir no arquivo editar.php e no formulário criar um campo oculto que vai ser <input type="hidden" name="id" value="<?$info['id'];?>"> com isso vamos mandar as 3 informações juntas para o editar_action.php.

Arquivo **editar_action.php**

```
<?php

//conexão com o banco de dados

require 'config.php';

$id = filter_input(INPUT_POST, 'id');

$name = filter_input(INPUT_POST, 'name');

$email = filter_input(INPUT_POST, 'email',FILTER_VALIDATE_EMAIL);


if($id && $name && $email){

  //UPDATE usuarios SET name = '...', email = '...' WHERE id = '';

  $sql = $pdo->prepare("UPDATE usuarios SET nome = :name, email = :email WHERE id = :id");

  $sql->bindvalue(':id', $id);

  $sql->bindvalue(':name', $name);

  $sql->bindvalue(':email', $email);

  $sql->execute();

  header("Location: index.php");

  exit;


} else {

   header("Location: adicionar.php");
```

```
    exit;
}
```

Agora é ir no index.php e clicar no botão do editar em algum dos registros e efetuar a alteração.

Delete: Removendo dados com PDO

Vamos fazer o processo de excluir um registro no banco de dados. Para isso vamos criar o arquivo editar.php. Esse processo é simples, clicando no botão excluir da página index.php vai para a página excluir.php, esse recebe, verifica e exclui.

Arquivo **excluir.php**

```
<?php
    require 'config.php';
    //1º passos

    $id = filter_input(INPUT_GET,'id'); //recebe via GET
    if($id){
        $sql = $pdo->prepare("DELETE FROM usuarios WHERE id = :id");
        $sql->bindvalue(':id', $id);
        $sql->execute();
    }
    header("Location: index.php");
    exit;
?>
```

Agora voltamos na página index.php e fizemos a verificação se está excluindo clicando no botão excluir em um registro.

Para evitar de excluimos um arquivo acidentalmente, vamos colocar no arquivo index.php um javascript que vai pedir uma confirmação se deseja realmente excluir o registro.

```
 <a href="excluir.php?id=<?=$usuario['id'];?>" onclick="return confirm('Tem certeza que deseja excluir?')">[Excluir]</a>
```

Até aqui vocês puderam ver todo o processo de um CRUD. Na próxima aula vamos ver mais observações sobre o CRUD.