

MANUAL
DE UTILIZAÇÃO
FRAMEWORK
BÁSICO
PHP-MVC

SUMÁRIO

1. PHP ORIENTADO A OBJETOS PARA INICIANTES.....	3
2. CONFIGURANDO O AMBIENTE	14
3. CRIANDO E CONFIGURANDO UMA BASE DE DADOS	18
4. CONFIGURANDO A CONEXÃO COM O BANCO DE DADOS	21
5. CRIANDO UM MODELO (MODEL)	23
6. CRIANDO O CONTROLE (CONTROL)	31
7. CONFIGURANDO O MENU	37
8. CRIANDO AS VISUALIZAÇÕES (VIEW)	39

1. PHP ORIENTADO A OBJETOS PARA INICIANTES

Para muitos programadores PHP, orientação a objetos é um conceito amedrontador, cheio de sintaxes complicadas e pontos de paradas, neste manual você aprenderá os conceitos por trás da programação orientada a objetos (POO), um estilo de codificação onde ações relacionadas são agrupadas em classes para ajudar na criação de códigos mais compactos e efetivos.

Programação orientada a objetos é um estilo de programação que permite os desenvolvedores agruparem tarefas semelhantes em classes. Isso ajuda a mantermo-nos dentro do princípio "don't repeat yourself" (DRY) (em português, não se repita), além de facilitar a manutenção do código. "Programação orientada a objetos é um estilo de programação que permite os desenvolvedores agrupar tarefas semelhantes em classes."

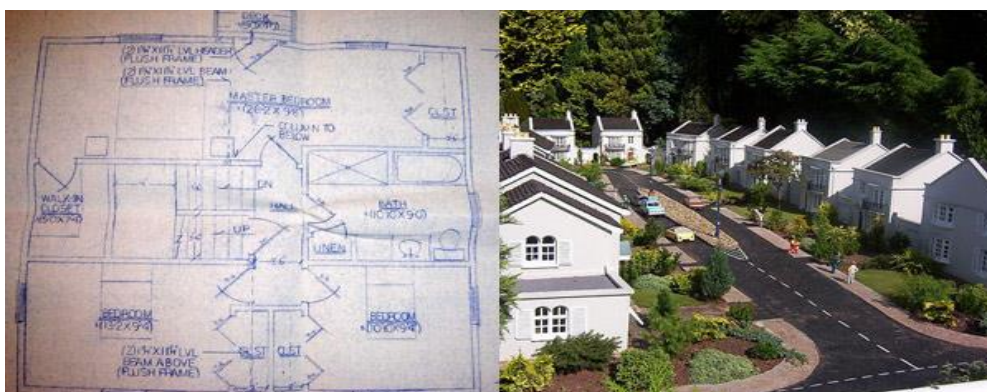
Um dos maiores benefícios da programação DRY é que, se alguma informação é alterada em seu programa, geralmente, só uma mudança é necessária para atualizar o código. Um dos maiores problemas para os desenvolvedores é ter de manter códigos onde os dados são declarados e redeclarados, acabando num jogo de pique esconde, em busca de funcionalidades e dados duplicados pelo código.

POO é intimidadora para uma grande quantidade de programadores porque ela introduz uma nova sintaxe e, a primeira vista, parece ser bem mais complexa que a simples programação procedural. Entretanto, se analisar bem, POO é bem direta e, fundamentalmente, uma abordagem mais simples para programação.

Compreendendo Objetos e Classes

Antes de lidar diretamente com as partes mais interessante da POO, um entendimento básico das diferenças entre **objetos** e **classes** é necessário. Essa seção falará sobre a base das classes, suas diferentes capacidades e alguns de seus usos.

Reconhecendo as Diferenças Entre Objetos e Classes



Os programadores de longa data falam sobre objetos e classes, esses termos parecem permutáveis entre si. Porém, não é bem assim.

Logo de cara, há uma grande confusão na POO: quando programadores de longa data falam sobre objetos e classes, esses termos parecem permutáveis entre si. Porém, não é bem assim, mesmo que a diferença entre eles seja um pouco complicada de perceber, no início.

Uma classe, por exemplo, é como uma **planta baixa para uma casa**. Ela define a forma da casa no papel, com as relações entre as diferentes partes da casa, claramente definidas e planejadas, mesmo a casa ainda não existindo.

Um objeto, por outro lado, **seria a casa de verdade**, construída de acordo com a planta baixa. Os dados guardados no objeto são como a madeira, fios e concreto que compoem a casa: sem a ordem criada pela planta baixa, são só um monte de materiais. Entretanto, quando tudo é colocado seguindo uma ordem, eles viram uma casa organizada de verdade e útil.

As classes servem de estrutura para os dados e ações, e usam essa informação para construir objetos. Mais de um objeto pode ser construído de uma mesma classe, ao mesmo tempo, cada um independente dos outros. Continuando a nossa analogia de classes e objetos em relação à construção, é parecida com a maneira que um condomínio de casas pode ser construído, usando a mesma planta baixa: 150 casas diferentes, todas bastante parecidas umas às outras, porém, contem famílias e decorações diferentes por dentro.

Estruturando Classes

A sintaxe para criar uma classe é bem direta: declare-a usando a palavra chave class, seguida do nome da classe e um par de chaves ({}):

```
<?php
class MyClass
{
    // As propriedades e métodos da Classe vem aqui
}
?>
```

Após criar a classe, ela pode ser instanciada e guardada em alguma variável usando a palavra chave new:

```
$obj = new MyClass;
```

Para vermos o conteúdo da classe, usamos var_dump():

```
var_dump($obj);
```

Experimente isso, colocando todo o código anterior em um único arquivo php, chamado test.php na sua pasta local de testes:

```

<?php
class MyClass
{
    // As propriedades e métodos da Classe vem aqui
}
$obj = new MyClass;

var_dump($obj);
?>

```

Carregue a página no seu navegador, usando o endereço `http://localhost/test.php` (ou o endereço indicado pelo seu servidor local) e o resultado a seguir deve aparecer:

```
object(MyClass)#1 (0) { }
```

De uma forma bem simplória, você acabou de criar seu primeiro código em POO.

Definindo as Propriedades da Classe

Para adicionar dados à classe, usamos as propriedades, que são variáveis específicas à classe. Elas funcionam de forma parecida às variáveis normais, exceto que elas estão ligadas ao objeto e só podem ser acessadas usando o objeto.

Para adicionar uma propriedade a `MyClass`, adicione o seguinte trecho de código ao seu script:

```

<?php
class MyClass
{
    public $prop1 = "Sou um propriedade de classe!";
}
$obj = new MyClass;
var_dump($obj);
?>

```

A palavra chave `public` determina a visibilidade da propriedade, a qual você aprenderá mais sobre, nas próximas seções. Depois disso, a propriedade é nomeada usando os padrões básicos de nomeação de variáveis, e, então, um valor é atribuído (embora propriedades de classe não necessitem de valores iniciais).

Para ver o valor da propriedade e mostrá-la no navegador, reference o objeto o qual será feita a leitura, bem como a propriedade a ser lida, dessa forma:

```
echo $obj->prop1;
```

Já que múltiplas instâncias de uma mesma classe podem existir, se um objeto em específico não for referenciado, o código não será capaz de determinar de qual objeto ler a propriedade. O uso da flecha (`->`) é um construto da POO no PHP que permite acessar as propriedades e métodos de um dado objeto.

Modifique o código do arquivo test.php para ler a propriedade ao invés de mostrar todo o conteúdo da classe, dessa forma:

```
<?php
class MyClass
{
    public $prop1 = "Sou um propriedade de classe!";
}

$obj = new MyClass;
echo $obj->prop1; // Mostra a saída/conteúdo da propriedade
?>
```

Atualizando a página no seu navegador, você obtém isso, agora:

Sou uma propriedade de classe!

Definindo Métodos de Classe

Métodos são funções específicas das classes. Ações particulares que os objetos serão capazes de executar são definidas dentro das classes na forma de métodos.

Por exemplo, para criar métodos que atribuam e retorne o valor de uma propriedade de classe chamada \$prop1, adicione o código a seguir:

```
<?php
class MyClass
{
    public $prop1 = "Sou um propriedade de classe!";

    public function setProperty($newval)
    {
        $this->prop1 = $newval;
    }
    public function getProperty()
    {
        return $this->prop1 . "<br />";
    }
}
$obj = new MyClass;
echo $obj->prop1;
?>
```

Nota — POO permite que os objetos referenciem-se usando \$this. Quando estiver dentro de um método, use \$this da mesma forma que você usaria o nome do objeto fora da classe.

Para usar os métodos, execute-os da mesma forma que faria com funções normais, mas, antes, referenceie o objeto ao qual eles pertencem. Para ler o valor da propriedade \$prop1 da classe MyClass, depois altera-lo e, por fim, le-lo novamente, faça as seguintes alterações no código:

```
<?php
class MyClass
{
    public $prop1 = "Sou uma propriedade de classe!";

    public function setProperty($newval)
    {
        $this->prop1 = $newval;
    }

    public function getProperty()
    {
        return $this->prop1 . "<br />";
    }
}

$obj = new MyClass;

echo $obj->getProperty(); // Lê o valor da propriedade
$obj->setProperty("Sou um novo valor da propriedade!"); // Atribui um novo valor
echo $obj->getProperty(); // Lê o valor novamente para mostrar a mudança
?>
```

Atualize a página no seu navegador e você verá o resultado a seguir:

Sou uma propriedade de classe!

Sou um novo valor da propriedade!

"O poder da POO mostra-se ao usar múltiplas instâncias da mesma classe."

```
<?php

class MyClass
{
    public $prop1 = "Sou uma propriedade de classe!";
```

```

public function setProperty($newval)
{
    $this->prop1 = $newval;
}

public function getProperty()
{
    return $this->prop1 . "<br />";
}
}

// Create two objects
$obj = new MyClass;
$obj2 = new MyClass;

// Mostra o valor de $prop1 de ambos os objetos
echo $obj->getProperty();
echo $obj2->getProperty();

// Atribui novos valores para ambos os objetos
$obj->setProperty("Sou um novo valor de propriedade!");
$obj2->setProperty("Pertencço à segunda instância!");

// Mostra o valor de $prop1 de ambos os objetos
echo $obj->getProperty();
echo $obj2->getProperty();

?>

```

Quando atualizar a página do seu navegador, verá o resultado a seguir:

Sou uma propriedade de classe!

Sou uma propriedade de classe!

Sou um novo valor de propriedade!

Pertencço à segunda instância!

Como pode ver, POO mantém os objetos como entidades diferentes, o que torna fácil a separação de diferentes partes de código em pedaços pequenos e relacionados.

Métodos Mágicos em POO

Para facilitar o uso dos objetos, o PHP provê uma série de métodos mágicos, métodos especiais chamados quando certas ações comuns ocorrem com objetos. Isso permite executar várias tarefas úteis com certa facilidade.

Usando Construtores e Destruidores

Quando um objeto é instanciado, é desejável que algumas coisas ocorram de cara. Para lidar com isso, o PHP provê o método `__construct()`, que é chamado automaticamente quando um novo objeto é criado.

Para ilustrar os conceitos dos construtores, adicione um construtor à classe `MyClass` que mostre uma mensagem toda vez que uma nova instância for criada:

```
<?php

class MyClass
{
    public $prop1 = "Sou uma propriedade de classe!";

    public function __construct()
    {
        echo 'A classe "', __CLASS__, '" foi instanciada!<br />';
    }

    public function setProperty($newval)
    {
        $this->prop1 = $newval;
    }

    public function getProperty()
    {
        return $this->prop1 . "<br />";
    }
}

// Cria um novo objeto
$obj = new MyClass;

// Mostra o valor de $prop1
echo $obj->getProperty();

// Mostra uma mensagem ao final do arquivo
echo "Fim do arquivo.<br />";

?>
```

Nota — `__CLASS__` retorna o nome da classe na qual foi usado; isso é o que chamamos de constante mágica. Há inúmeras constantes mágicas disponíveis e você pode ler mais sobre elas no manual do PHP.

Atualize a página no seu navegador e terá os resultados a seguir:

A classe "MyClass" foi instanciada!

Sou uma propriedade de classe!

Fim do arquivo.

Para chamar uma função quando um objeto for destruído, o método mágico `__destruct()` está disponível. Ele é útil para finalizar as tarefas da classe (encerrar uma conexão com a base de dados, por exemplo).

Mostre uma mensagem quando um objeto for destruído usando o método mágico

`__destruct()` na class MyClass:

```
<?php

class MyClass
{
    public $prop1 = "Sou uma propriedade de classe!";

    public function __construct()
    {
        echo 'A classe "', __CLASS__, '" foi instanciada!<br />';
    }

    public function __destruct()
    {
        echo 'A classe "', __CLASS__, '" foi destruída.<br />';
    }

    public function setProperty($newval)
    {
        $this->prop1 = $newval;
    }

    public function getProperty()
    {
        return $this->prop1 . "<br />";
    }
}

// Cria um novo objeto
$obj = new MyClass;

// Mostra o valor de $prop1
echo $obj->getProperty();

// Mostra uma mensagem ao final do arquivo
echo "Fim do arquivo.<br />";

?>
```

Com um método destruidor definido, atualize a página no seu navegador e verá o resultado a seguir:

A classe "MyClass" foi instanciada!

Sou uma propriedade de classe!

Fim do arquivo.

A classe "MyClass" foi destruída.

"Quando o fim do arquivo é alcançado, o PHP libera, automaticamente, todos os recursos."

Para executar, explicitamente, o método destruidor, você pode destruir o objeto usando a função

unset():

```
<?php

class MyClass
{
    public $prop1 = "Sou uma propriedade de classe!";

    public function __construct()
    {
        echo 'A classe "', __CLASS__, "' foi instanciada!<br />';
    }

    public function __destruct()
    {
        echo 'A classe "', __CLASS__, "' foi destruída.<br />';
    }

    public function setProperty($newval)
    {
        $this->prop1 = $newval;
    }

    public function getProperty()
    {
        return $this->prop1 . "<br />";
    }
}

// Cria um novo objeto
$obj = new MyClass;

// Mostra o valor de $prop1
```

```
echo $obj->getProperty();

// Destrói o objeto
unset($obj);

// Mostra uma mensagem ao final do arquivo
echo "Fim do arquivo.<br />";

?>
```

Os resultados das mudanças aparecerão assim que você atualizar a página no seu navegador:

A classe "MyClass" foi instanciada!

Sou uma propriedade de classe!

A classe "MyClass" foi destruída.

Fim do arquivo.

Usando Herança de Classe

Classes podem herdar métodos e propriedades de outra classe usando a palavra chave extends. Por exemplo, para criar uma segunda classe que estenda MyClass e adicione um outro método, você faria dessa forma:

```
<?php

class MyClass
{
    public $prop1 = "Sou uma propriedade de classe!";

    public function __construct()
    {
        echo 'A classe "', __CLASS__, "' foi instanciada!<br />';
    }

    public function __destruct()
    {
        echo 'A classe "', __CLASS__, "' foi destruída.<br />';
    }

    public function __toString()
    {
        echo "Usando o método toString: ";
        return $this->getProperty();
    }
}
```

```

public function setProperty($newval)
{
    $this->prop1 = $newval;
}

public function getProperty()
{
    return $this->prop1 . "<br />";
}
}

class MyOtherClass extends MyClass
{
    public function newMethod()
    {
        echo "De um novo método na classe " . __CLASS__ . "<br />";
    }
}

// Cria um novo objeto
$newobj = new MyOtherClass;

// Usa o método da nova classe
echo $newobj->newMethod();

// Usa um método da classe pai
echo $newobj->getProperty();

?>

```

Após atualizar a página no seu navegador, você terá o seguinte resultado:

A classe "MyClass" foi instanciada!

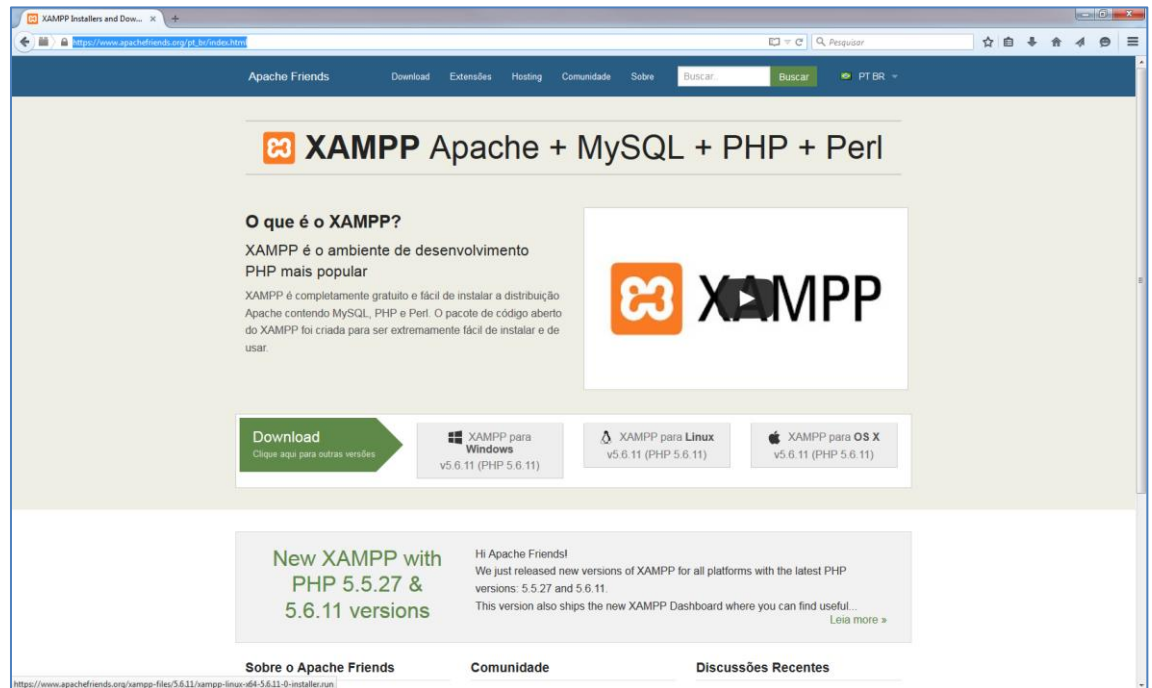
De um novo método na classe MyOtherClass.

Sou uma propriedade de classe!

A classe "MyClass" foi destruída.

2. CONFIGURANDO O AMBIENTE

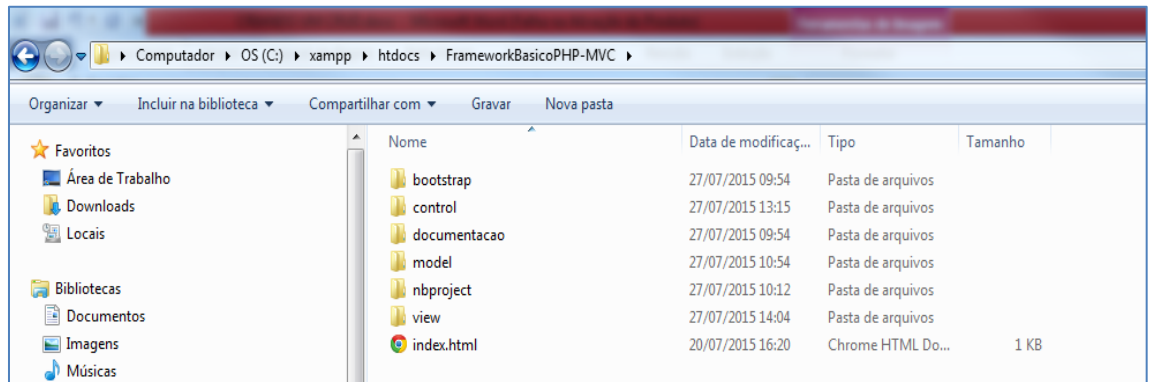
a) Baixe e instale o XAMPP (https://www.apachefriends.org/pt_br/index.html)



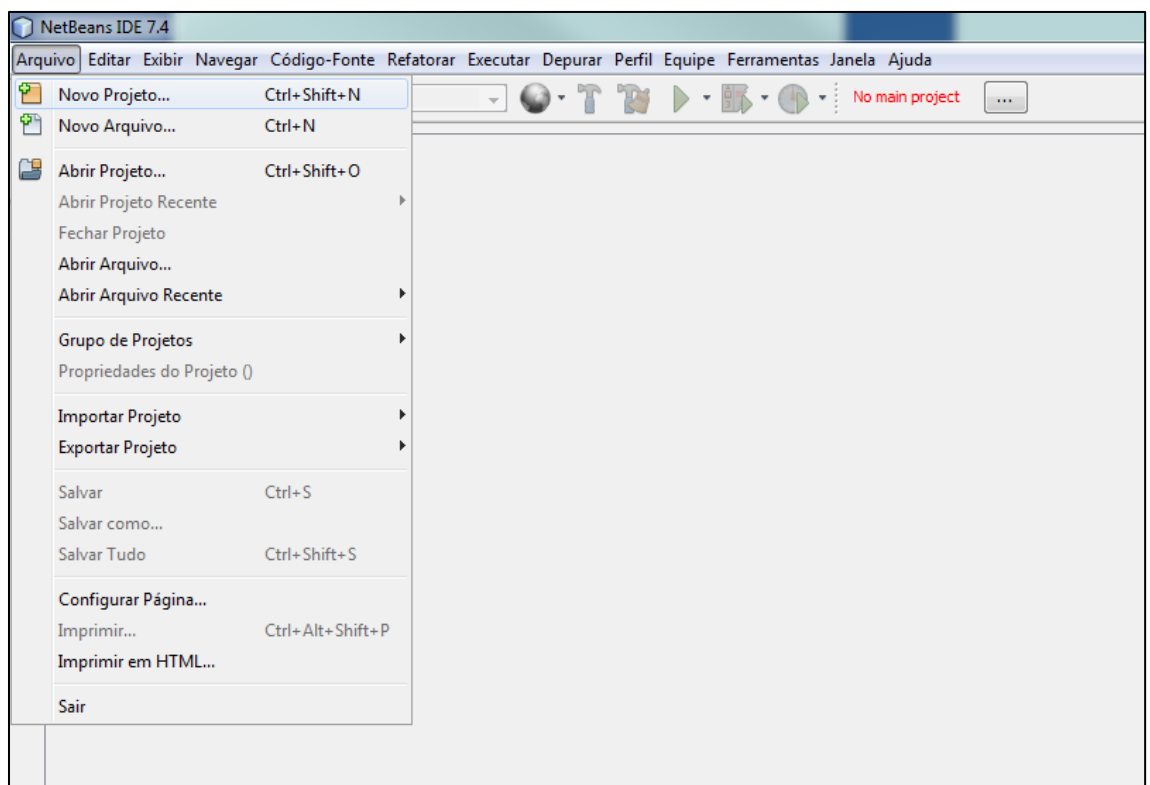
b) Baixe e instale o Netbeans IDE PHP (<https://netbeans.org/downloads/>)



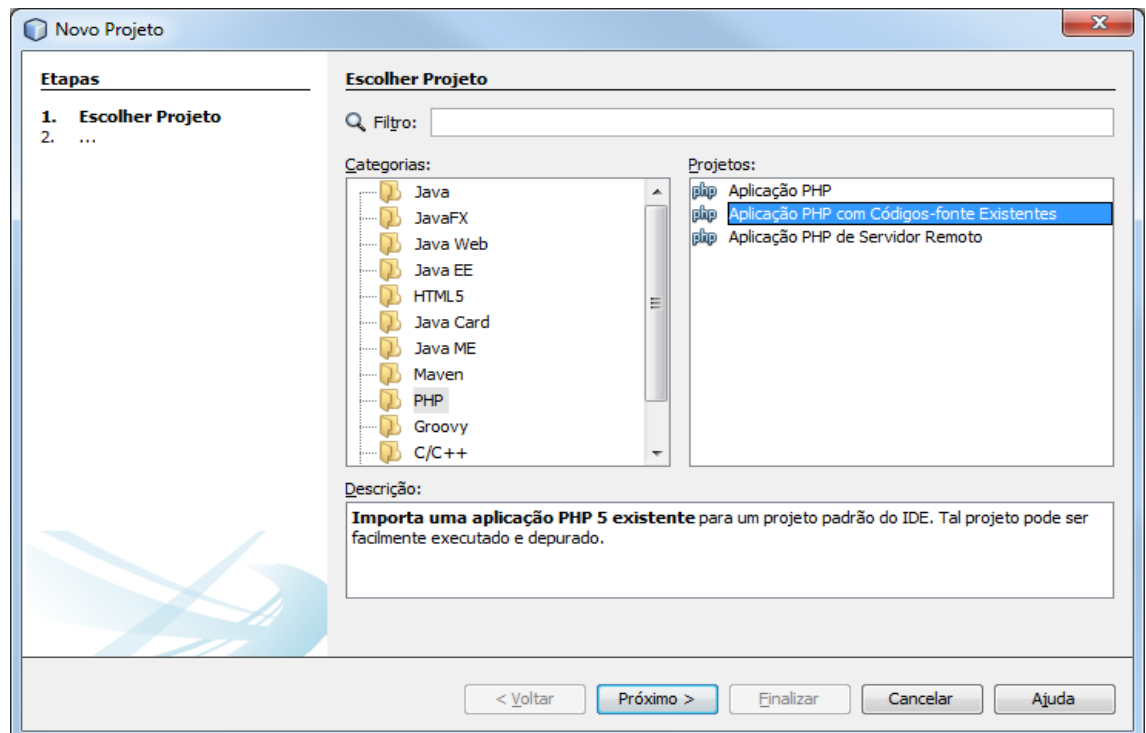
- c) Copie a pasta do Framework para: C:\xampp\htdocs\Framework Básico PHP-MVC e renomeie para *FrameworkBasicoPHP-MVC* (retirar os espaços e acentos)



- d) Abra o Netbeans IDE e crie um novo projeto



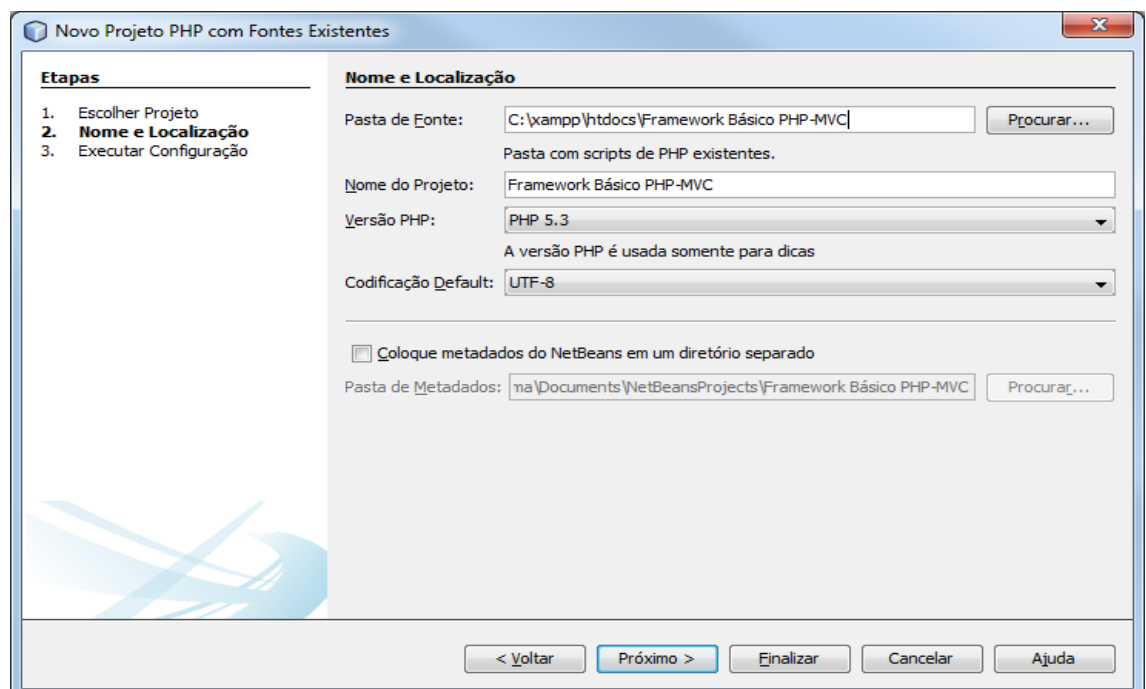
- e) Escolha a categoria PHP e projetos Aplicação PHP com Códigos-fonte Existentes



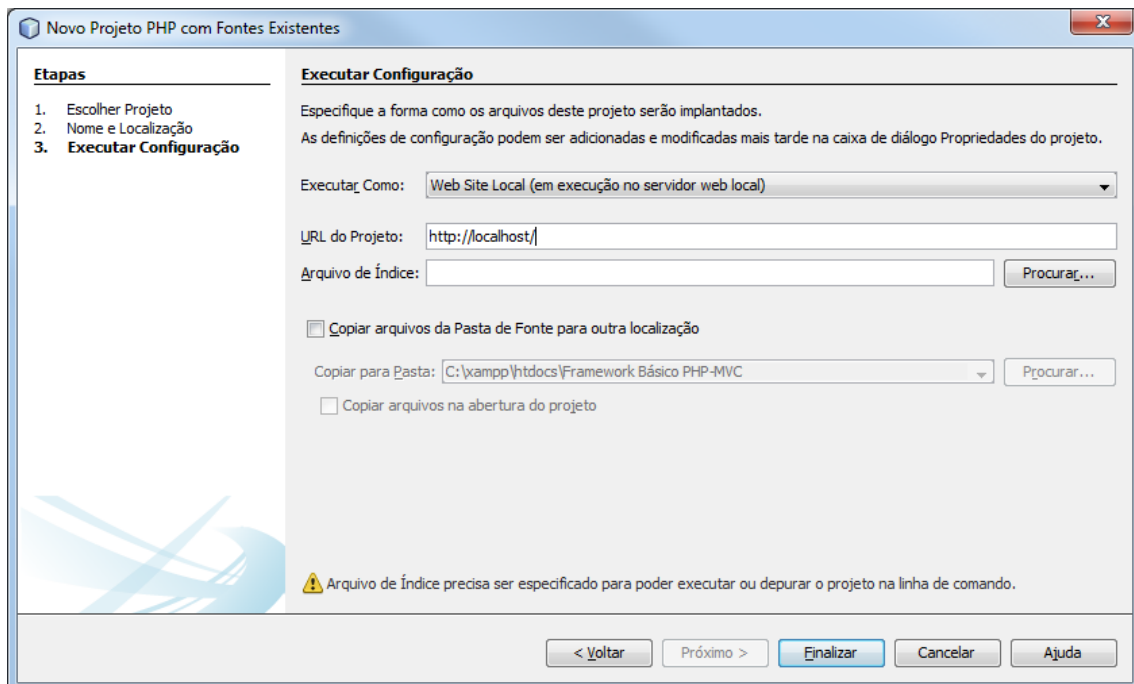
- f) Indique o caminho e o nome do projeto:

Pasta de Fonte: C:\xampp\htdocs\FrameworkBasicoPHP-MVC

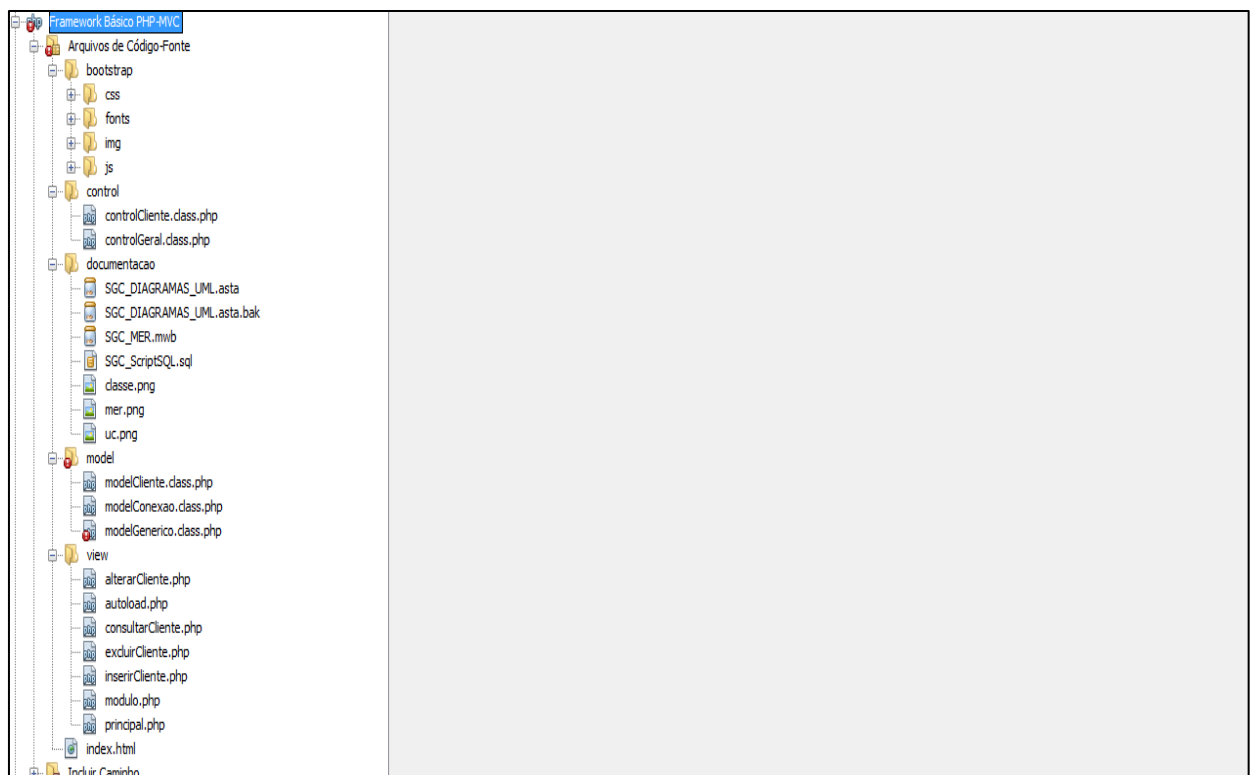
Nome do Projeto: Framework Básico PHP-MVC (ou o nome que desejar)



g) Configure a URL do projeto e clique em finalizar

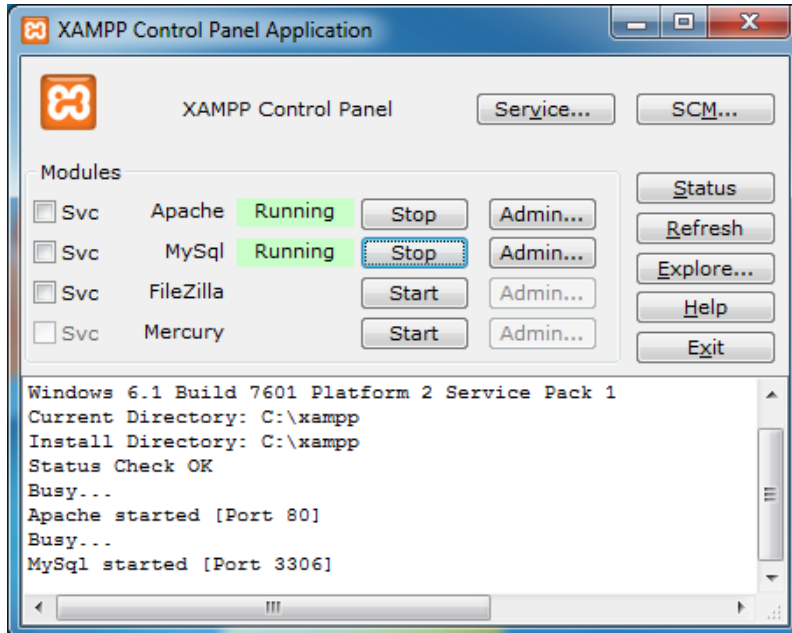


h) Abra as pastas do projeto no NetBeans IDE.

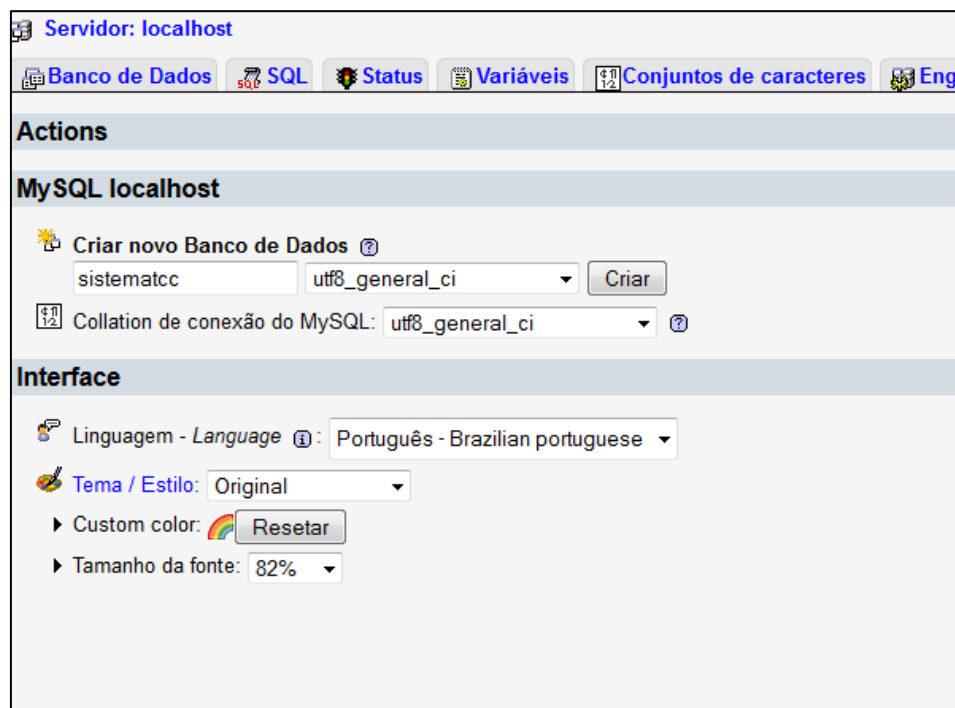


3. CRIANDO E CONFIGURANDO UMA BASE DE DADOS

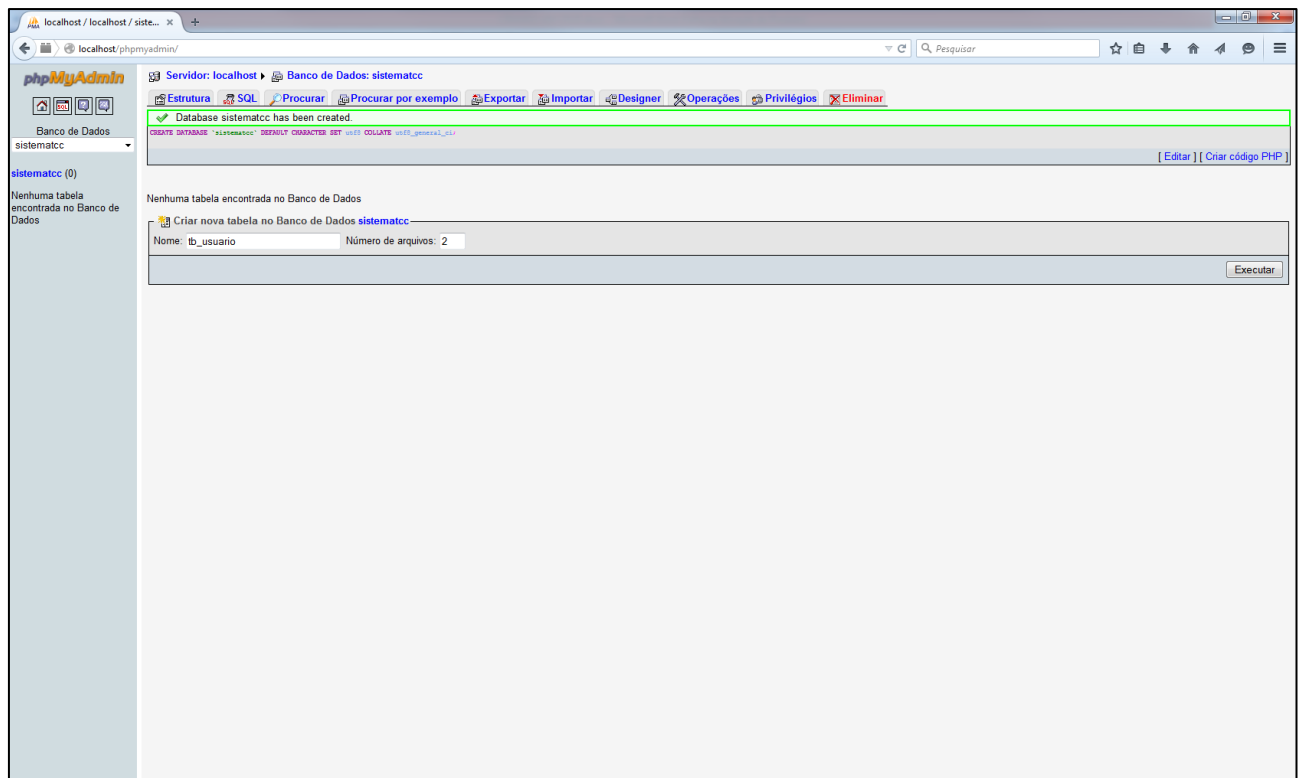
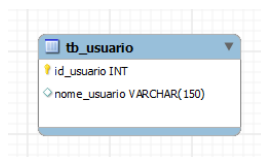
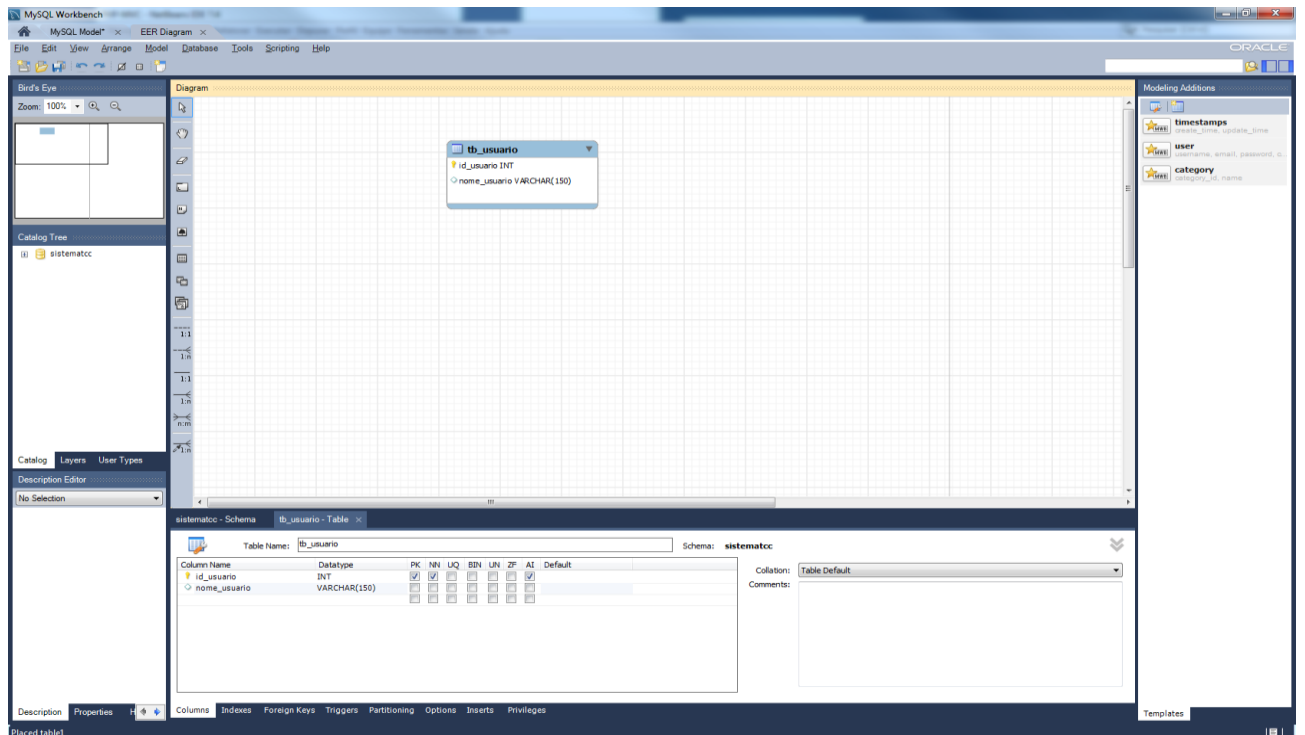
a) Inicie o XAMPP e acesse o phpmyadmin: <http://localhost/phpmyadmin/>



b) Crie uma base de dados com o nome que desejar



c) Crie uma tabela conforme o modelo abaixo: (foi utilizado o MySQL Workbench)



localhost / localhost / siste...

localhost/phpmyadmin/Pesquisar

phpMyAdmin

Banco de Dados

sistematcc

sistematcc (0)

Nenhuma tabela encontrada no Banco de Dados

Servidor: localhostBanco de Dados: sistematccTabela: tb_usuario

Campo	id_usuario	nome_usuario
Tipo	INT	VARCHAR
Tamanho/Definir	150	
Padrão	Nenhum	Nenhum
Collation		
Atributos		
Nulo		
Índice	PRIMARY	
AUTO_INCREMENT		
Comentários		
MIME-type		
Transformações do navegador		
Opções de transformação		

Comentários da tabela:

Storage Engine: MyISAMCollation:

PARTITION definition:

SalvarOu Adicionar 1 campo(s)Executar

¹ Se um tipo de campo é "enum" ou "set", por favor entre valores usando este formato: 'a','b','c'.
Se você for colocar uma barra contrária ("") ou aspas simples (") entre os valores, coloque uma barra contrária antes (por exemplo 'lyyz' ou 'a\b').

² Para valores padrão, digite um valor simples, sem barras de escape ou aspas, use este formato: a

³ Digite os valores para as opções de transformação usando este formato: 'a', 100, b'c'...

Se por acaso precisar inserir uma contra-barra ("\") ou aspas (") no meio desses valores, faça-o usando outra contra-barra 'lyyz' ou 'a\b').
Para uma lista de opções de transformação disponíveis e suas transformações MIME-type, clique em [descrição de transformações](#)

localhost / localhost / siste...

localhost/phpmyadmin/Pesquisar

phpMyAdmin

Banco de Dados

sistematcc (1)

sistematcc (1)

tb_usuario

Servidor: localhostBanco de Dados: sistematccTabela: tb_usuario

VisualizarEstruturaSQLProcurarInserirExportarImportarOperaçõesLimparEliminar

Table 'sistematcc`.`tb_usuario' has been created.

CREATE TABLE `sistematcc`.`tb_usuario`
(`id_usuario` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`nome_usuario` VARCHAR(150) NOT NULL
ENGINE = MYISAM)

[Editar][Criar código PHP]

	Campo	Tipo	Collation	Atributos	Nulo	Padrão	Extra	Ação
<input type="checkbox"/>	id_usuario	int(11)			Não	Nenhum	auto_increment	<div></div>
<input type="checkbox"/>	nome_usuario	varchar(150)	utf8_general_ci		Não	Nenhum		<div></div>

Marcar todos / Desmarcar todosCom marcados:

Visualização para impressãoVer relaçõesPropor estrutura da tabela

Adicionar 1 campo(s)No final da tabelaNo início da tabelaDepois id_usuarioExecutar

Índices:

Ação	Nome chave	Tipo	Único	Packed	Campo	Cardinalidade	Collation	Nulo	Comment
<div></div>	PRIMARY	BTREE	Sim	Não	id_usuario	0	A		

Criar um índice em 1 colunasExecutar

Uso do espaço

Estatísticas do registros

Tipo	Uso	Comandos	Valor
Dados	Bytes		
Índice	Bytes		
Total	Bytes		

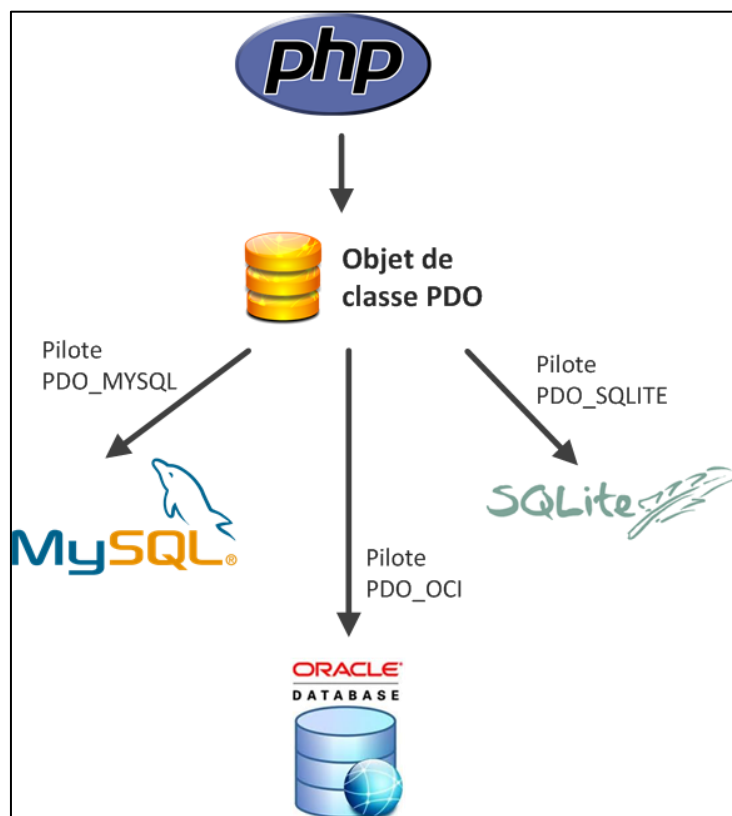
4. CONFIGURANDO A CONEXÃO COM O BANCO DE DADOS

Neste Framework é utilizado para conexão com o banco de dados MySQL com o PHP PDO. O PDO PHP Data Object (PDO) é um módulo de PHP montado sob o paradigma Orientado a Objetos e cujo objetivo é prover uma padronização da forma com que PHP se comunica com um banco de dados relacional. Este módulo surgiu a partir da versão 5 de PHP. O PDO, portanto, é uma interface que define um conjunto de classes e a assinatura dos métodos de comunicação com uma base de dados.

Antes de iniciar a utilização do PDO habilite as linhas abaixo no seu arquivo *C:\xampp\php\php.ini*, geralmente eles estão comentados.

Habilitando PDO no Windows

```
extension=php_pdo.dll  
extension=php_pdo_mysql.dll  
extension=php_pdo_pgsql.dll  
extension=php_pdo_sqlite.dll
```



a) No Netbeans acesso o projeto, abra a pasta model e edite *modelConexao.class.php* e faça as configurações conforme seu ambiente de desenvolvimento



```

public function conectar() {

    #setar as configurações do banco de dados
    $this->setHost("localhost");
    $this->setUser("root");
    $this->setSenha("");
    $this->setDbase("sistemacc");

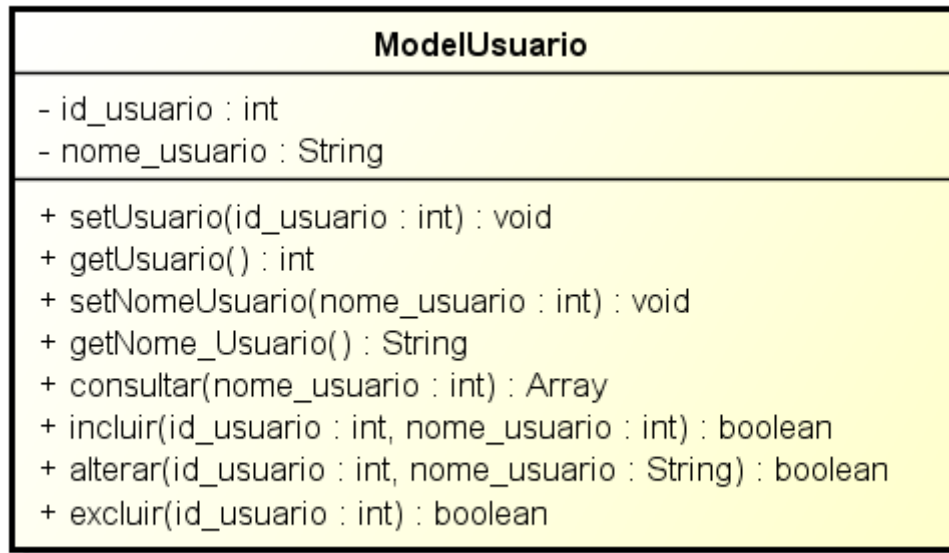
    #conecta ao banco de dados usando o PHP PDO
    try {
        $pdo = new PDO("mysql:host={$this->getHost()};dbname={$this->getDbase()}", "{$this->getUser()}", "{$this->getSenha()}");
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $this->setLink($pdo);
        return $pdo;
    } catch (PDOException $e) {
        $this->setLink(null);
        return false;
    }
}

```

5. CRIANDO UM MODELO (MODEL)

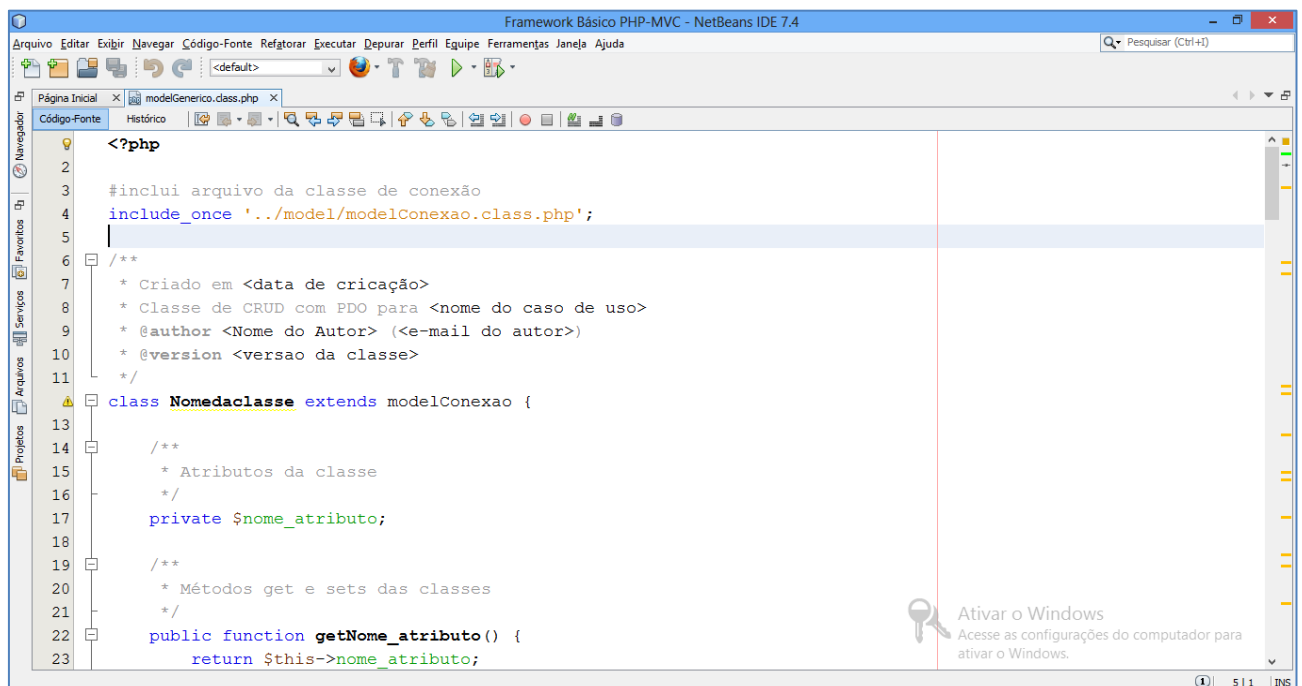
O modelo (model) consiste nos dados da aplicação, regras de negócios, lógica e funções.

a) o modelo deve ser baseado no diagrama de classe, abaixo segue o modelo que será criado.



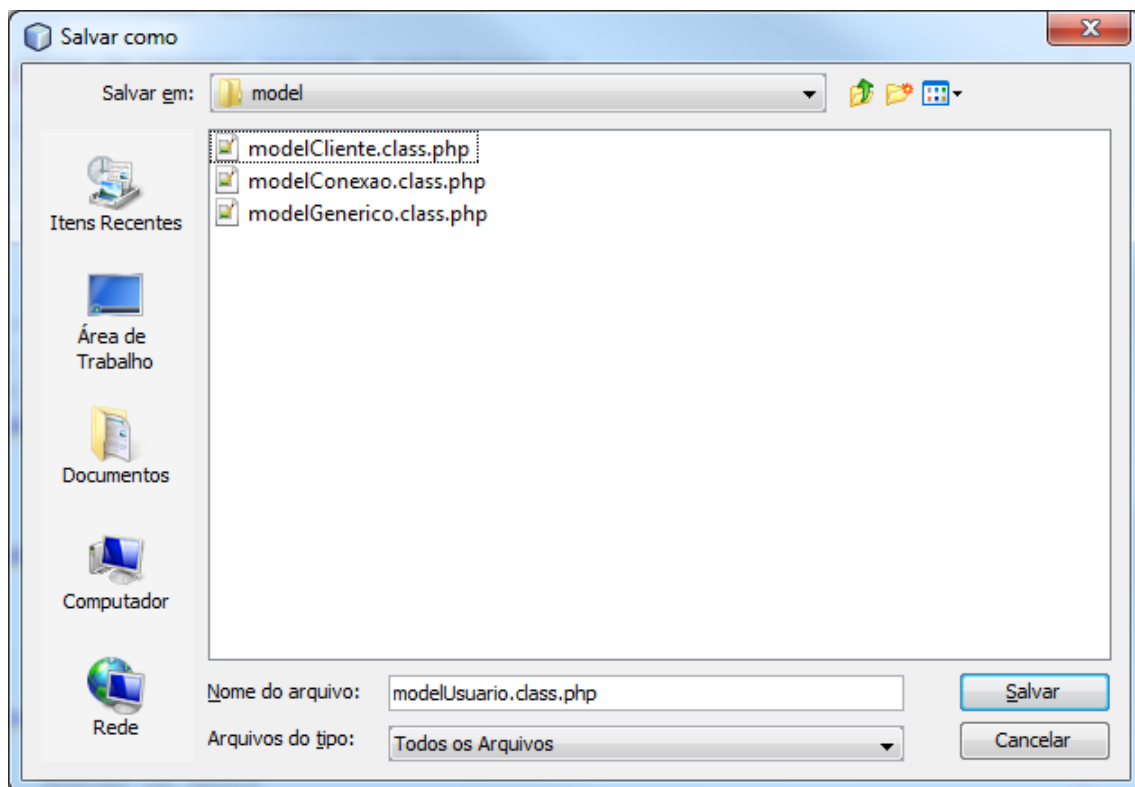
powered by Astah

b) No Netbeans, na pasta modelo, abra o arquivo *modeloGenerico.class.php*



```
<?php
2
3 #incluir arquivo da classe de conexão
4 include_once '../model/modelConexao.class.php';
5
6 /**
7  * Criado em <data de criação>
8  * Classe de CRUD com PDO para <nome do caso de uso>
9  * @author <Nome do Autor> (<e-mail do autor>)
10  * @version <versao da classe>
11  */
12
13 class Nomedaclassa extends modelConexao {
14
15     /**
16      * Atributos da classe
17      */
18     private $nome_atributo;
19
20     /**
21      * Métodos get e sets das classes
22      */
23     public function getNome_atributo() {
24         return $this->nome_atributo;
25     }
26 }
```

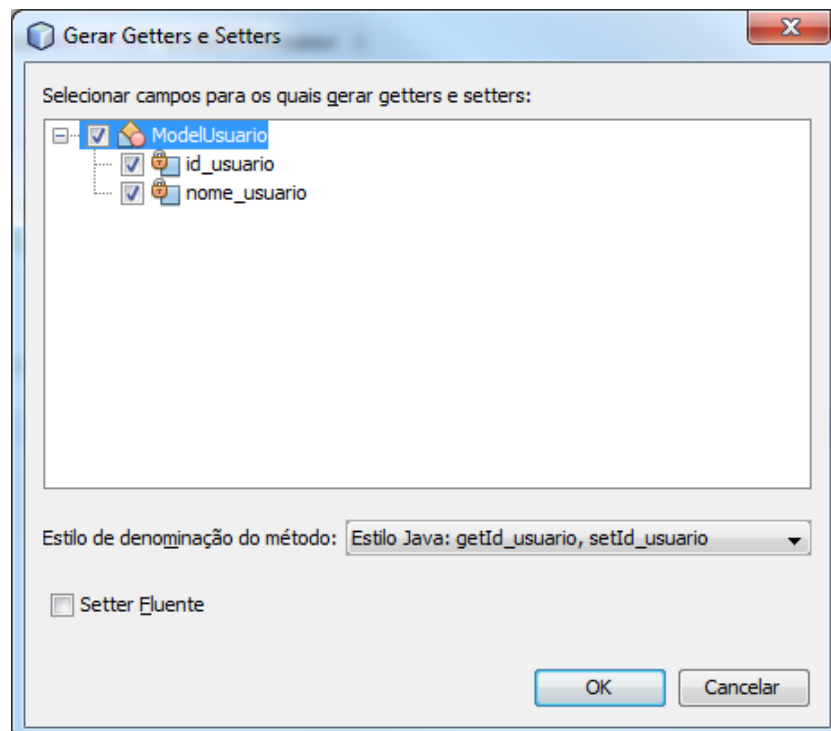
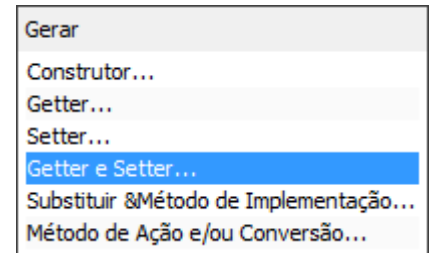
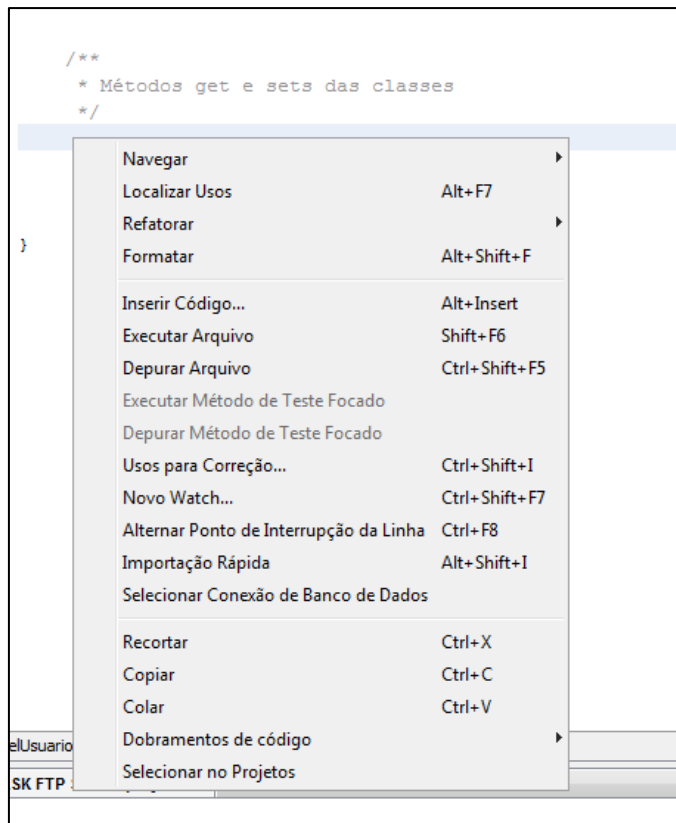
c) Salve este arquivo como: *modelUsuario.class.php* indo em Arquivo|Salvar Como...



d) Configure o nome da classe e os atributos

```
<?php
2
3 #inclui arquivo da classe de conexão
4 include_once '../model/modelConexao.class.php';
5
6 /**
7  * Criado em <data de criação>
8  * Classe de CRUD com PDO para <nome do caso de uso>
9  * @author <Nome do Autor> (<e-mail do autor>)
10  * @version <versao da classe>
11  */
12 class ModelUsuario extends modelConexao {
13
14     /**
15      * Atributos da classe
16      */
17     private $id_usuario;
18     private $nome_usuario;
19
20
21     /**
```


e) Crie os métodos Getter e Setter. O Netbeans cria estes métodos automaticamente clicando com o botão direito do mouse e escolhendo a opção Inserir Código | Getter e Setter



```

/**
 * Métodos get e sets das classes
 */
public function getId_usuario() {
    return $this->id_usuario;
}

public function getNome_usuario() {
    return $this->nome_usuario;
}

public function setId_usuario($id_usuario) {
    $this->id_usuario = $id_usuario;
}

public function setNome_usuario($nome_usuario) {
    $this->nome_usuario = $nome_usuario;
}

```

f) Criando o método *consultarUsuario*

```

public function consultarUSuario($nome_usuario) {

    #setar os dados
    $this->setNome_usuario($nome_usuario);

    #montar a consultar (where true serve para selecionar todos os registros)
    $sql = "select * from tb_usuario where true";

    #verificar se foi passado algum valor de <parâmetro>
    if ($this->getNome_usuario() != null) {
        $sql.= " and nome_usuario=:nome_usuario";
    }

    #executa consulta e controla um array com o resultado da consulta
    try {
        $bd = $this->conectar();
        $query = $bd->prepare($sql);

        #verificar se foi passado algum valor de :<nome campo>
        if ($this->getNome_usuario() != null) {
            $query->bindValue(':nome_usuario', $this->getNome_usuario(), PDO::PARAM_STR);
        }
        $query->execute();
        $this->resultado = $query->fetchAll(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {

        $this->resultado = null;
    }

    return $this->resultado;
}

```

Código-fonte:

```
public function consultarUSuario($id_usuario, $nome_usuario) {

    #setar os dados
    $this->setNome_usuario($nome_usuario);
    $this->setId_usuario($id_usuario);

    #montar a consultar (where true serve para selecionar todos os registros)
    $sql = "select * from tb_usuario where true";

    #verificar se foi passado algum valor de nome do usuario
    if ($this->getNome_usuario() != null) {
        $sql.= " and nome_usuario=:nome_usuario";
    }

    #verificar se foi passado algum valor de id do usuario
    if ($this->getId_usuario() != null) {
        $sql.= " and id_usuario=:id_usuario";
    }

    #executa consulta e controli um array com o resultado da consulta
    try {
        $bd = $this->conectar();
        $query = $bd->prepare($sql);

        #verificar se foi passado algum valor de nome do usuario
        if ($this->getNome_usuario() != null) {
            $query->bindValue(':nome_usuario', $this->getNome_usuario(), PDO::PARAM_SRT);
        }

        #verificar se foi passado algum valor de id do usuario
        if ($this->getId_usuario() != null) {
            $query->bindValue(':id_usuario', $this->getId_usuario(), PDO::PARAM_INT);
        }

        $query->execute();
        $this->resultado = $query->fetchAll(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {

        $this->resultado = null;
    }
    return $this->resultado;
}
```

g) Criando o método *inserirUsuario*

```
function inserirUsuario($id_usuario,$nome_usuario) {

    #setar os dados
    $this->setId_usuario(null);
    $this->setNome_usuario($nome_usuario);

    #montar a consulta
    $sql = "INSERT INTO tb_usuario(id_usuario,nome_usuario) VALUES (null,:nome_usuario)";

    #realizar a blidagem dos dados
    try {
        $bd = $this->conectar();
        $query = $bd->prepare($sql);
        $query->bindValue(':nome_usuario', $this->getNome_usuario(), PDO::PARAM_STR);
        $query->execute();
        return true;
    } catch (PDOException $e) {
        return false;
    }
}
```

Código fonte:

```
function inserirUsuario($nome_usuario) {

    #setar os dados
    $this->setId_usuario(null);
    $this->setNome_usuario($nome_usuario);

    #montar a consulta
    $sql = "INSERT INTO tb_usuario(id_usuario,nome_usuario) VALUES (null,:nome_usuario)";

    #realizar a blidagem dos dados
    try {
        $bd = $this->conectar();
        $query = $bd->prepare($sql);
        $query->bindValue(':nome_usuario', $this->getNome_usuario(), PDO::PARAM_STR);
        $query->execute();
        return true;
    } catch (PDOException $e) {
        echo $e->getMessage();
        return false;
    }
}
```

h) Criando o método *alterarUsuario*

```
function alterarUsuario($id_usuario,$nome_usuario) {

    #setar os dados
    $this->setId_usuario($id_usuario);
    $this->setNome_usuario($nome_usuario);

    #montar a consulta
    $sql = "UPDATE tb_usuario SET nome_usuario = :nome_usuario WHERE id_usuario = :id_usuario";

    #realizar a blidagem dos dados
    try {
        $bd = $this->conectar();
        $query = $bd->prepare($sql);
        $query->bindValue(':id_usuario', $this->getId_usuario(), PDO::PARAM_INT);
        $query->bindValue(':nome_usuario', $this->getNome_usuario(), PDO::PARAM_SRT);
        $query->execute();
        return true;
    } catch (PDOException $e) {

        return false;
    }
}
```

Código-fonte:

```
function alterarUsuario($id_usuario, $nome_usuario) {

    #setar os dados
    $this->setId_usuario($id_usuario);
    $this->setNome_usuario($nome_usuario);

    #montar a consulta
    $sql = "UPDATE tb_usuario SET nome_usuario = :nome_usuario WHERE id_usuario = :id_usuario";

    #realizar a blidagem dos dados
    try {
        $bd = $this->conectar();
        $query = $bd->prepare($sql);
        $query->bindValue(':id_usuario', $this->getId_usuario(), PDO::PARAM_INT);
        $query->bindValue(':nome_usuario', $this->getNome_usuario(), PDO::PARAM_SRT);
        $query->execute();
        return true;
    } catch (PDOException $e) {

        return false;
    }
}
```

i) Criando o método *excluirUsuario*

```
function excluirUsuario($id_usuario) {  
  
    #setar os dados  
    $this->setId_usuario($id_usuario);  
  
    #montar a consulta  
    $sql = "DELETE FROM tb_usuario WHERE id_usuario = :id_usuario";  
  
    #realizar a blidagem dos dados  
    try {  
        $bd = $this->conectar();  
        $query = $bd->prepare($sql);  
        $query->bindValue(':id_usuario', $this->getId_usuario(), PDO::PARAM_INT);  
        $query->execute();  
        return true;  
    } catch (PDOException $e) {  
        return false;  
    }  
}
```

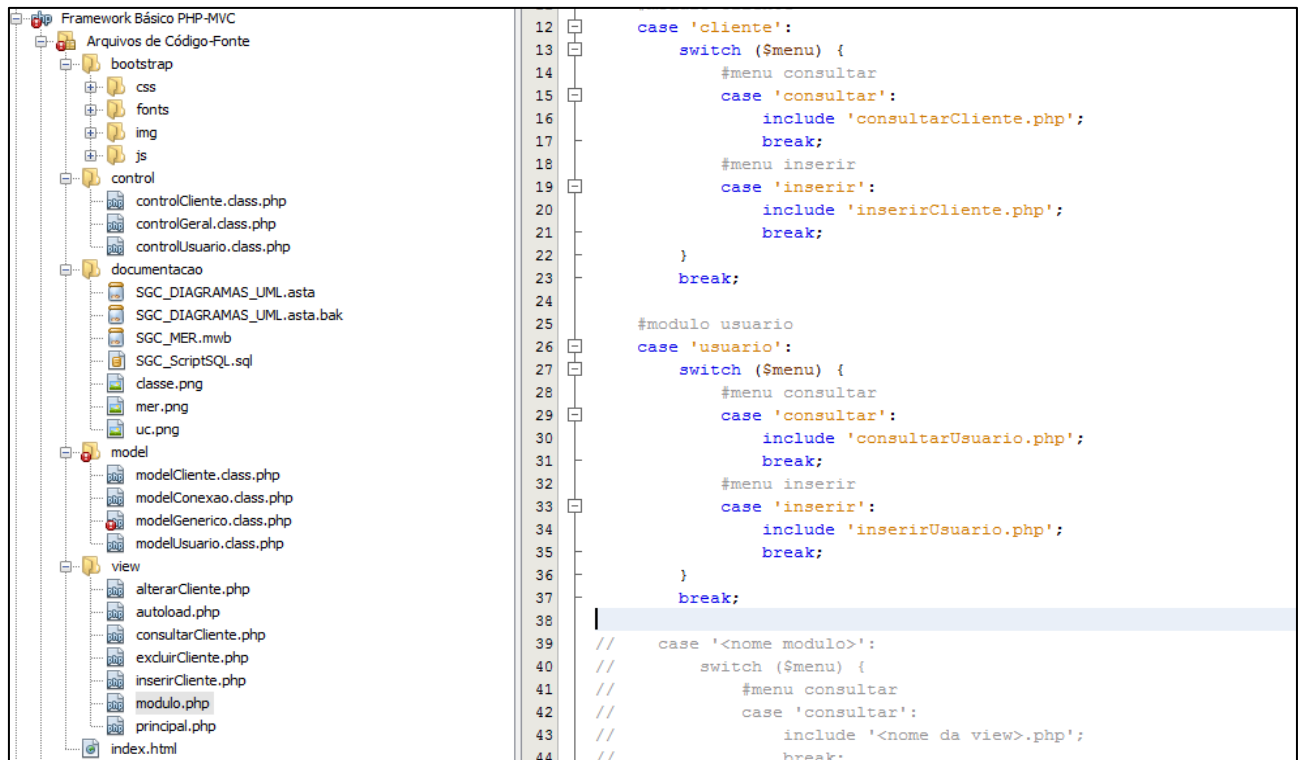
Código-fonte:

```
function excluirUsuario($id_usuario) {  
  
    #setar os dados  
    $this->setId_usuario($id_usuario);  
  
    #montar a consulta  
    $sql = "DELETE FROM tb_usuario WHERE id_usuario = :id_usuario";  
  
    #realizar a blidagem dos dados  
    try {  
        $bd = $this->conectar();  
        $query = $bd->prepare($sql);  
        $query->bindValue(':id_usuario', $this->getId_usuario(), PDO::PARAM_INT);  
        $query->execute();  
        return true;  
    } catch (PDOException $e) {  
        return false;  
    }  
}
```

6. CRIANDO O CONTROLE (CONTROL)

O controlador (controller) faz a mediação da entrada, convertendo-a em comandos para o modelo ou visão.

a) Configurar o *modulo.php*. Este arquivo php serve para realizar os redirecionamentos do sistema. Para criar um novo modulo do sistema basta coloca-lo dentro do bloco switch{} do php. O redirecionamento é realizado no controle



```
12 case 'cliente':
13     switch ($menu) {
14         #menu consultar
15         case 'consultar':
16             include 'consultarCliente.php';
17             break;
18         #menu inserir
19         case 'inserir':
20             include 'inserirCliente.php';
21             break;
22     }
23     break;
24
25 #modulo usuario
26 case 'usuario':
27     switch ($menu) {
28         #menu consultar
29         case 'consultar':
30             include 'consultarUsuario.php';
31             break;
32         #menu inserir
33         case 'inserir':
34             include 'inserirUsuario.php';
35             break;
36     }
37     break;
38
39 // case '<nome modulo>':
40 //     switch ($menu) {
41 //         #menu consultar
42 //         case 'consultar':
43 //             include '<nome da view>.php';
44 //             break;
```

```
#modulo usuario
case 'usuario':
    switch ($menu) {
        #menu consultar
        case 'consultar':
            include 'consultarUsuario.php';
            break;
        #menu inserir
        case 'inserir':
            include 'inserirUsuario.php';
            break;
    }
    break;
```

Código-fonte:

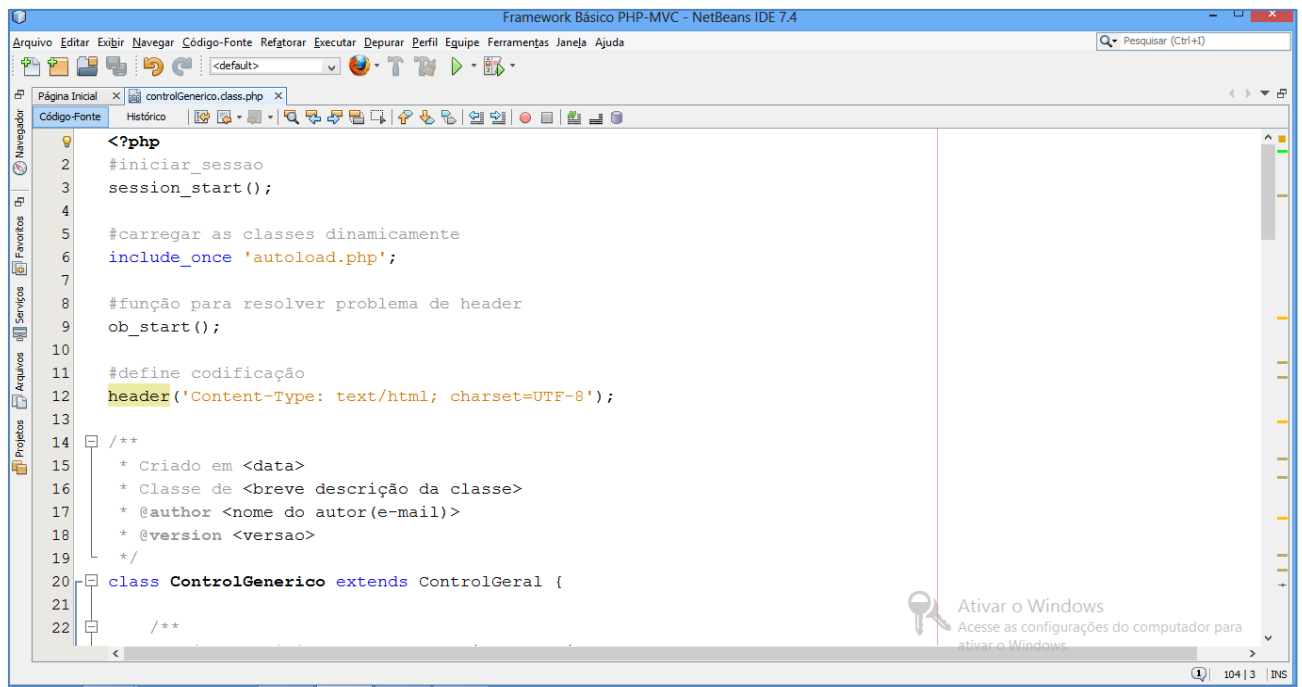
```
switch ($modulo) {
    #modulo cliente
    case 'cliente':
        switch ($menu) {
            #menu consultar
            case 'consultar':
                include 'consultarCliente.php';
                break;
            #menu inserir
            case 'inserir':
                include 'inserirCliente.php';
                break;
        }
        break;

    #modulo usuario
    case 'usuario':
        switch ($menu) {
            #menu consultar
            case 'consultar':
                include 'consultarAluno.php';
                break;
            #menu inserir
            case 'inserir':
                include 'inserirAluno.php';
                break;
        }
        break;

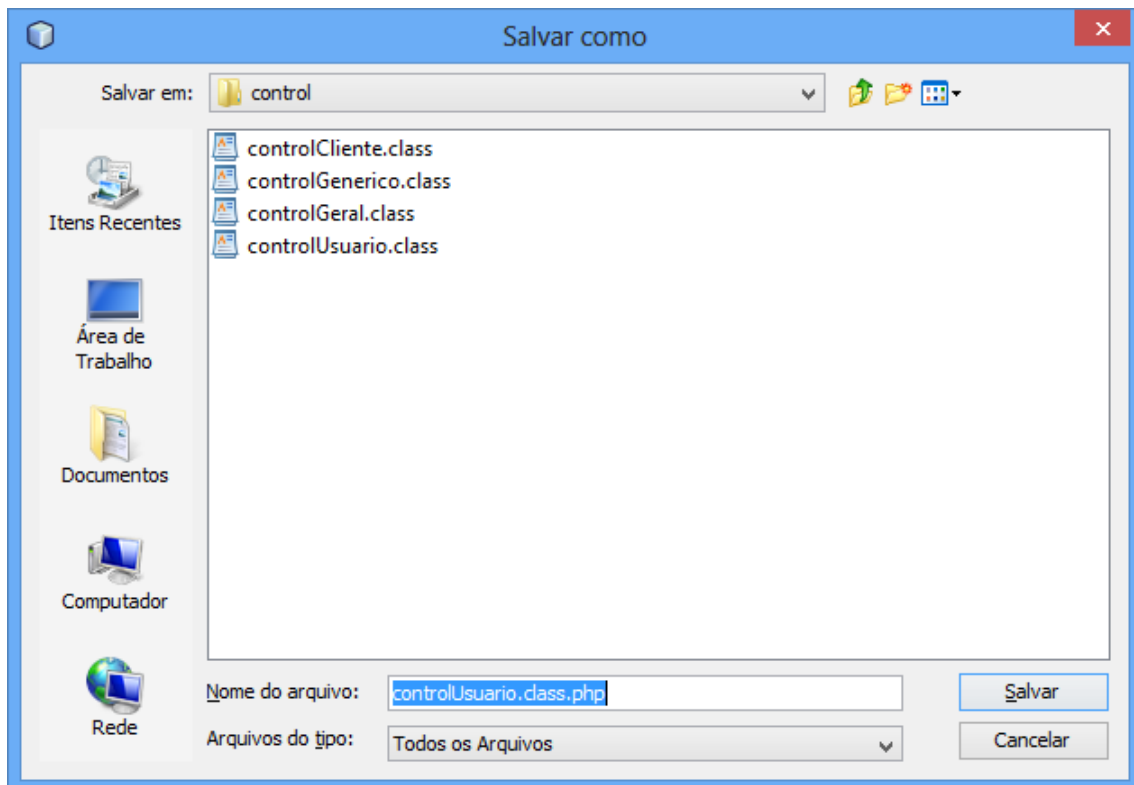
    // case '<nome modulo>':
    //     switch ($menu) {
    //         #menu consultar
    //         case 'consultar':
    //             include '<nome da view>.php';
    //             break;
    //         #menu inserir
    //         case 'inserir':
    //             include '<nome da view>.php';
    //             break;
    //     }
    //     break;

    default:
        #menu padrão
        include 'principal.php';
        break;
}
```


b) No Netbeans, na pasta control, abra o arquivo controlGenerico.class.php



c) Salve este arquivo como: *controlUsuario.class.php* indo em Arquivo|Salvar Como...



c) Crie os métodos de controle e faça os redirecionamentos

Consultar

```
/**
 * Método utilizado para validar os dados dos usuarios cadastrados e invocar o método consultarUsuario no model
 * @access public
 * @param String $nome nome do usuario
 * @return Array dados do usuario
 */
function consultar($nome_usuario) {

    #invocar método e passar parâmetros
    $objUsuario = new modelUsuario();
    return $listaUsuario = $objUsuario->consultarUsuario($id, $nome);
}
```

```
function consultar($id_usuario, $nome_usuario) {
```

```
    #invocar método e passar parâmetros
```

```
    $objUsuario = new modelUsuario();
```

```
    return $listaUsuario = $objUsuario->consultarUsuario($id_usuario, $nome_usuario);
```

```
}
```

Inserir

```
/**
 * Método utilizado para validar os dados dos usuarios cadastrados e invocar o método inserirUsuario no model
 * @access public
 * @param String $nome nome do usuario
 * @return Boolean retorna TRUE se os dados forem salvos com sucesso
 */
function inserir($nome_usuario) {

    #invocar método e passar parâmetros
    $objUsuario = new modelUsuario();

    #se for válido invocar o método de inserir
    if ($objUsuario->inserirUsuario($nome_usuario) == true) {
        #se for inserido com sucesso mostrar a mensagem
        $_SESSION['msg'] = "Inserido com sucesso!";
        #redirecionar
        header("location: ../view/modulo.php?modulo=usuario&menu=consultar");
    } else {
        $_SESSION['msg'] = "Erro ao inserir!";
        #redirecionar
        header("location: ../view/modulo.php?modulo=usuario&menu=consultar");
    }
}
```

```
function inserir($nome_usuario) {
```

```
    #invocar método e passar parâmetros
```

```
    $objUsuario = new modelUsuario();
```

```
    #se for válido invocar o método de inserir
```

```
    if ($objUsuario->inserirUsuario($nome_usuario) == true) {
```

```
        #se for inserido com sucesso mostrar a mensagem
```

```
        $_SESSION['msg'] = "Inserido com sucesso!";
```

```
        #redirecionar
```

```
        header("location: ../view/modulo.php?modulo=usuario&menu=consultar");
```

```
    } else {
```

```
        $_SESSION['msg'] = "Erro ao inserir!";
```

```
        #redirecionar
```

```
        header("location: ../view/modulo.php?modulo=usuario&menu=consultar");
```

```
    }
```

```
}
```

Alterar

```
/**
 * Método utilizado validar os dados dos usuarios e invocar o método alterarUsuario no mode
 * @access public
 * @param Int $id id do usuario
 * @return Boolean retorna TRUE se os dados forem salvos com sucesso
 */
function alterar($id_usuario, $nome_usuario) {

    #invocar método e passar parâmetros
    $objUsuario = new modelUsuario();

    if ($objUsuario->alterarUsuario($id_usuario, $nome_usuario) == true) {
        #se for alterado com sucesso mostrar a mensagem
        $_SESSION['msg'] = "Alterado com sucesso!";
        #redirecionar
        header("location: ../view/modulo.php?modulo=usuario&menu=consultar");
    } else {
        $_SESSION['msg'] = "Erro ao alterar!";
        #redirecionar
        header("location: ../view/modulo.php?modulo=usuario&menu=consultar");
    }
}
```

```
function alterar($id_usuario, $nome_usuario) {

    #invocar método e passar parâmetros
    $objUsuario = new modelUsuario();

    if ($objUsuario->alterarUsuario($id_usuario, $nome_usuario) == true) {
        #se for alterado com sucesso mostrar a mensagem
        $_SESSION['msg'] = "Alterado com sucesso!";
        #redirecionar
        header("location: ../view/modulo.php?modulo=usuario&menu=consultar");
    } else {
        $_SESSION['msg'] = "Erro ao alterar!";
        #redirecionar
        header("location: ../view/modulo.php?modulo=usuario&menu=consultar");
    }
}
```

Excluir

```
/**
 * Método utilizado para validar os dados dos usuarios e invocar o método excluirUsuario no model
 * @access public
 * @param Int $id id do usuario
 * @return Boolean retorna TRUE se os dados for excluído sucesso
 */
function excluir($id) {

    #invocar método e passar parâmetros
    $objUsuario = new modelUsuario();

    #invocar método e passar parâmetros
    if ($objUsuario->excluirUsuario($id_usuario) == true) {
        #se for excluído com sucesso mostrar a mensagem e redirecionar
        $_SESSION['msg'] = "Excluído com sucesso!";
        header("location: ../view/modulo.php?modulo=usuario&menu=consultar");
    } else {
        $_SESSION['msg'] = "Erro ao excluir!";
        header("location: ../view/modulo.php?modulo=usuario&menu=consultar");
    }
}
```

```
function excluir($id_usuario) {

    #invocar método e passar parâmetros
    $objUsuario = new modelUsuario();

    #invocar método e passar parâmetros
    if ($objUsuario->excluirUsuario($id_usuario) == true) {
        #se for excluído com sucesso mostrar a mensagem e redirecionar
        $_SESSION['msg'] = "Excluído com sucesso!";
        header("location: ../view/modulo.php?modulo=usuario&menu=consultar");
    } else {
        $_SESSION['msg'] = "Erro ao excluir!";
        header("location: ../view/modulo.php?modulo=usuario&menu=consultar");
    }
}
```

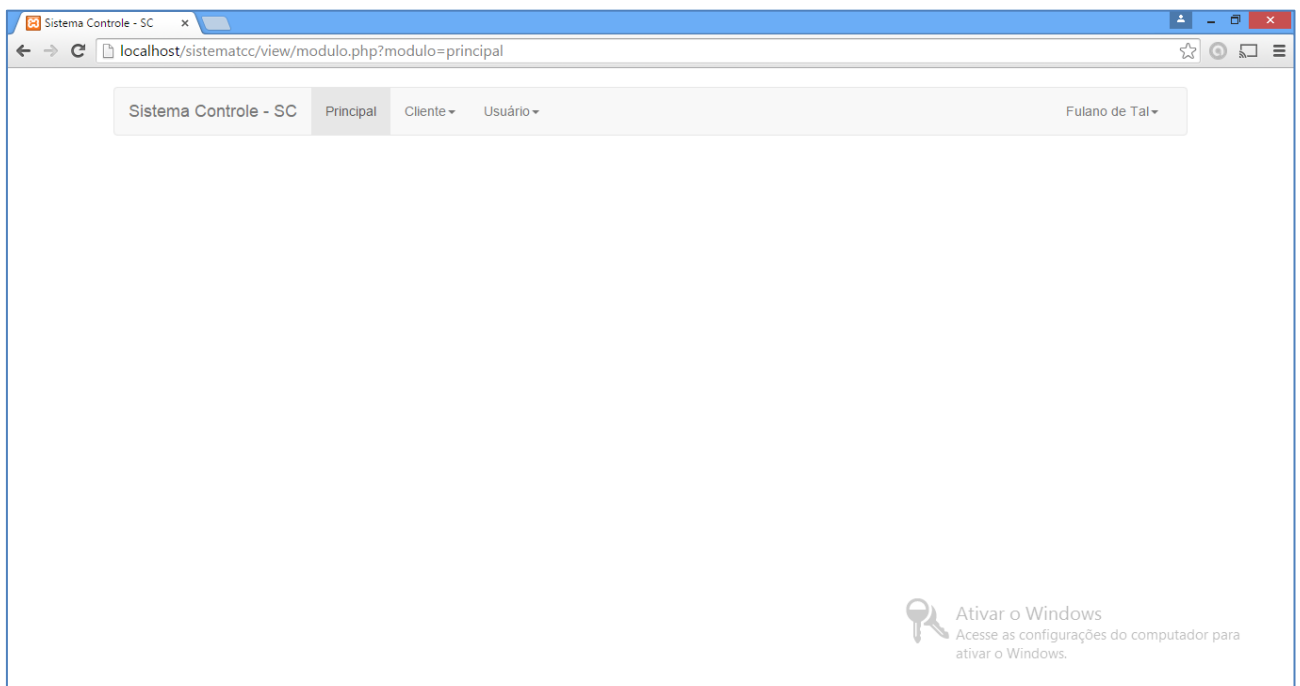
7. CONFIGURANDO O MENU

Para configurar o menu, acesse o controle *controlGeral.class.php* no método menu. É utilizado o Bootstrap para a montagem do menu.

a) Definir o nome do sistema. Coloque o nome que julgar adequado ao projeto

```
/**
 * Método utilizado para mostrar o menu do sistema
 * @access public
 * @param String $nomeSistema nome do sistema a ser exibido
 */
function menu($nomeSistema = 'Sistema Controle - SC') {
    echo' <!--Static navbar -->';
    echo' <nav class = "navbar navbar-default">';
    echo' <div class = "container-fluid">';
    echo' <div class = "navbar-header">';
    echo' <button type = "button" class = "navbar-toggle collapsed" data
    echo' <span class = "sr-only"></span>';
    echo' <span class = "icon-bar"></span>';
    echo' <span class = "icon-bar"></span>';
    echo' <span class = "icon-bar"></span>';
    echo' </div>';
```

Para verificar acesse: <http://localhost/FrameworkBasicoPHP-MVC/view/modulo.php?modulo=principal>



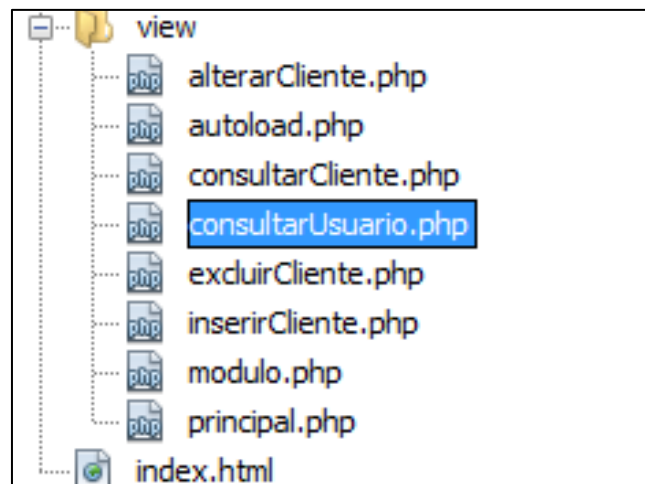
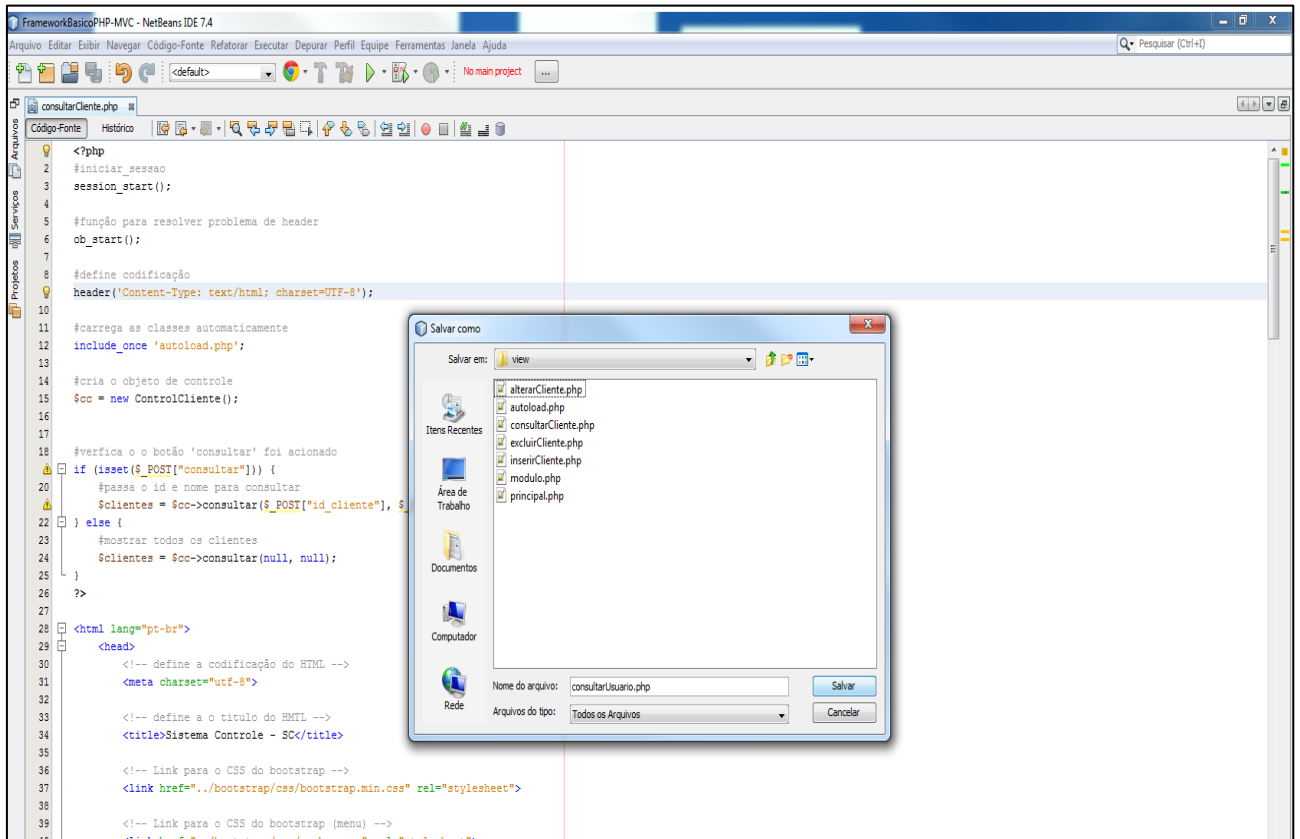
b) Criando um novo menu para Usuário, usando o seguinte código:

```
#inicio menu usuário
echo' <li class = "dropdown">';
echo' <a href = "#" class = "dropdown-toggle" data-toggle = "dropdown" role = "button"
aria-haspopup = "true" aria-expanded = "false">Usuário<span class = "caret"></span></a>';
echo' <ul class = "dropdown-menu">';
echo' <li><a href = "modulo.php?modulo=usuario&menu=consultar"><i class="icon-large
icon-search"></i>Consultar</a></li>';
echo' <li><a href = "modulo.php?modulo=usuario&menu=insrerir">Inserir</a></li>';
echo' </ul>';
echo' </li>';
#fim menu cliente
```



8. CRIANDO AS VISUALIZAÇÕES (VIEW)

a) Criando a view Consultar Usuário. Abra uma view já existente *consultarCliente.php* e salve como *consultarUsuario.php*



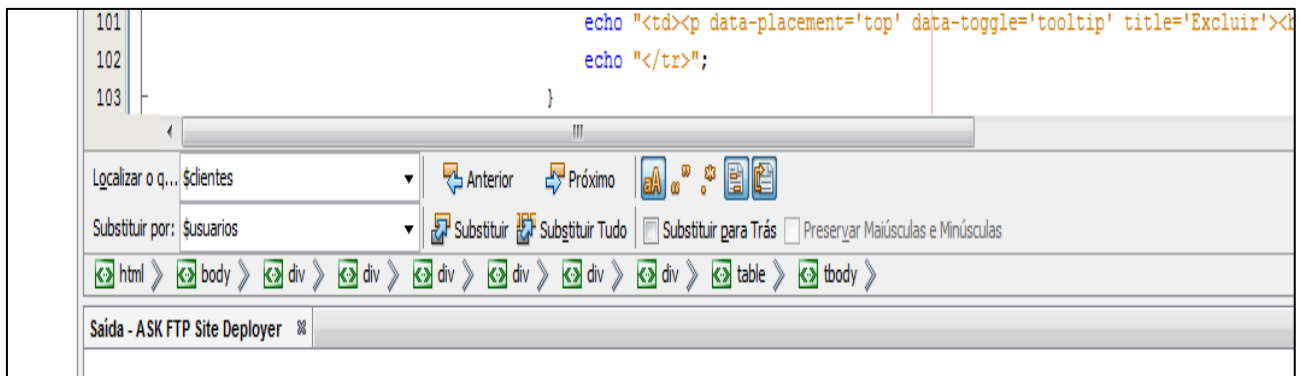
b) Substituição de nomes: No Netbeans acesse o menu Editar | Substituir...

Substituir: Cliente por Usuario



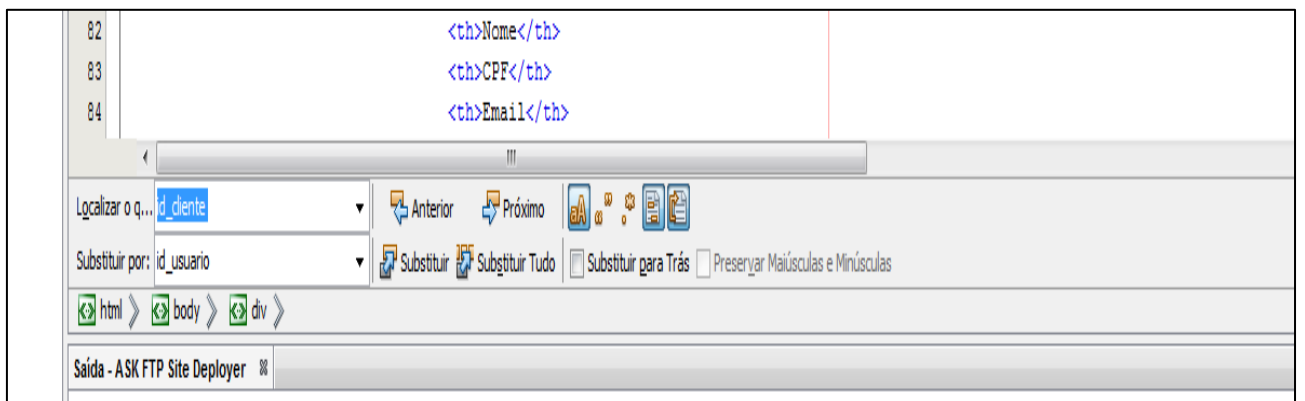
Clicar em substituir tudo

Substituir: \$clientes por \$usuarios



Clicar em substituir tudo

Substituir: id_cliente por id_usuario

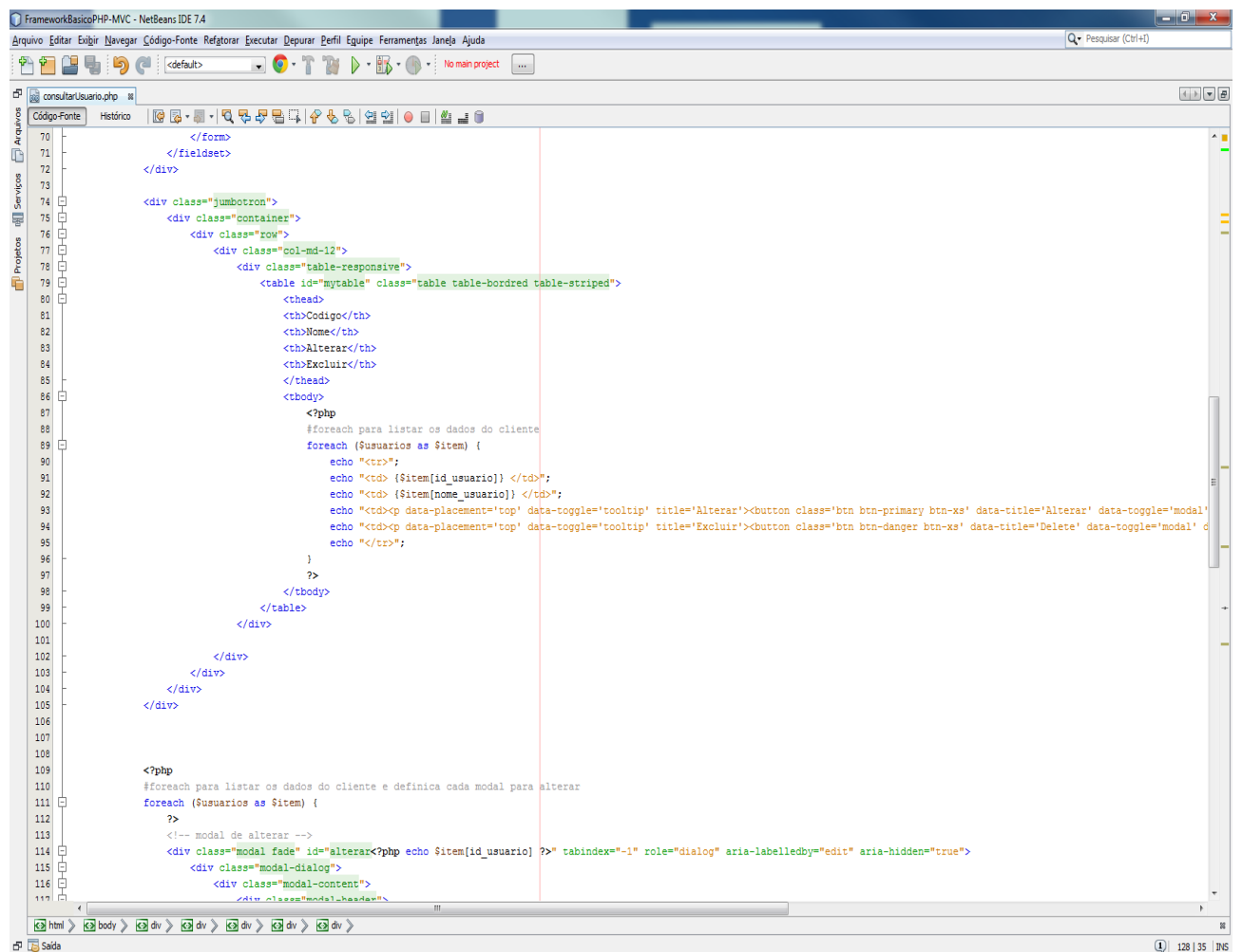


Clicar em substituir tudo

c) Definir o formulário de consulta em *consultarUsuario.php*

```
<!-- Main component for a primary marketing message or call to action -->
<div class="jumbotron">
  <fieldset>
    <legend>Dados da Consulta</legend>
    <form method="post">
      <div class="form-group">
        <label for="codigo" class="control-label">Código</label>
        <input type="text" class="form-control" placeholder="código" name="id_cliente">
        <label for="nome" class="control-label">Nome</label>
        <input type="text" class="form-control" placeholder="nome" name="nome">
      </div>
      <button type="submit" class="btn btn-primary" name="consultar" style="width: 100%;">Consultar</button>
    </form>
  </fieldset>
</div>
```

d) Definir o retorno da consulta em *consultarUsuario.php*



Código-fonte

```
<?php
#iniciar_sessao
session_start();

#função para resolver problema de header
ob_start();

#define codificação
header('Content-Type: text/html; charset=UTF-8');

#carrega as classes automaticamente
include_once 'autoload.php';

#cria o objeto de controle
$cc = new ControlUsuario();

#verifica o o botão 'consultar' foi acionado
if (isset($_POST["consultar"])) {
    #passa o id e nome para consultar
    $usuarios = $cc->consultar($_POST["id_usuario"], $_POST["nome_usuario"]);
} else {
    #mostrar todos os usuarios
    $usuarios = $cc->consultar(null, null);
}
?>

<html lang="pt-br">
<head>
    <!-- define a codificação do HTML -->
    <meta charset="utf-8">

    <!-- define a o titulo do HMTL -->
    <title>Sistema Controle - SC</title>

    <!-- Link para o CSS do bootstrap -->
    <link href="../../bootstrap/css/bootstrap.min.css" rel="stylesheet">

    <!-- Link para o CSS do bootstrap (menu) -->
    <link href="../../bootstrap/css/navbar.css" rel="stylesheet">

</head>
<body>

    <!-- Link para o JQuery do bootstrap -->
    <script src="../../bootstrap/js/jquery.min.js"></script>
    <script src="../../bootstrap/js/bootstrap.min.js"></script>
    <script src="../../bootstrap/js/ie10-viewport-bug-workaround.js"></script>

    <div class="container">
        <!-- inserir o menu -->
        <?php
            #mostrar o menu
            $cc->menu();
            $cc->alerta($_SESSION['msg']);
```

```

?>

<!-- Main component for a primary marketing message or call to action -->
<div class="jumbotron">
  <fieldset>
    <legend>Dados da Consulta</legend>
    <form method="post">
      <div class="form-group">
        <label for="codigo" class="control-label">Código</label>
        <input type="text" class="form-control" placeholder="código" name="id_usuario">
        <label for="nome" class="control-label">Nome</label>
        <input type="text" class="form-control" placeholder="nome" name="nome_usuario">
      </div>
      <button type="submit" class="btn btn-primary" name="consultar" style="width:
100%;">Consultar</button>
    </form>
  </fieldset>
</div>

<div class="jumbotron">
  <div class="container">
    <div class="row">
      <div class="col-md-12">
        <div class="table-responsive">
          <table id="mytable" class="table table-bordered table-striped">
            <thead>
              <th>Codigo</th>
              <th>Nome</th>
              <th>Alterar</th>
              <th>Excluir</th>
            </thead>
            <tbody>
              <?php
                #foreach para listar os dados do usuario
                foreach ($usuarios as $item) {
                  echo "<tr>";
                  echo "<td> {$item[id_usuario]} </td>";
                  echo "<td> {$item[nome_usuario]} </td>";
                  echo "<td><p data-placement='top' data-toggle='tooltip' title='Alterar'><button
class='btn btn-primary btn-xs' data-title='Alterar' data-toggle='modal' data-
target='#alterar{$item[id_usuario]}'><span class='glyphicon glyphicon-
pencil'></span></button></p></td>";
                  echo "<td><p data-placement='top' data-toggle='tooltip' title='Excluir'><button
class='btn btn-danger btn-xs' data-title='Delete' data-toggle='modal' data-
target='#excluir{$item[id_usuario]}'><span class='glyphicon glyphicon-
trash'></span></button></p></td>";
                  echo "</tr>";
                }
              ?>
            </tbody>
          </table>
        </div>
      </div>
    </div>
  </div>
</div>

```

```

<?php
#foreach para listar os dados do usuario e definica cada modal para alterar
foreach ($usuarios as $item) {
    ?>
    <!-- modal de alterar -->
    <div class="modal fade" id="alterar<?php echo $item[id_usuario] ?>" tabindex="-1"
role="dialog" aria-labelledby="edit" aria-hidden="true">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <button type="button" class="close" data-dismiss="modal" aria-hidden="true"><span
class="glyphicon glyphicon-remove" aria-hidden="true"></span></button>
                    <h4 class="modal-title custom_align" id="Heading">Alterar Usuario</h4>
                </div>
                <div class="modal-body">

                    <?php
                    #inclui a view alterar usuario
                    include 'alterarUsuario.php';
                    ?>

                </div>
            </div>
        </div>
    </div>
    <?php
}
?>

<?php
#foreach para listar os dados do usuario e definica cada modal para alterar
foreach ($usuarios as $item) {
    ?>
    <!-- modal de excluir -->
    <div class="modal fade" id="excluir<?php echo $item[id_usuario] ?>" tabindex="-1"
role="dialog" aria-labelledby="edit" aria-hidden="true">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <button type="button" class="close" data-dismiss="modal" aria-hidden="true"><span
class="glyphicon glyphicon-remove" aria-hidden="true"></span></button>
                    <h4 class="modal-title custom_align" id="Heading">Excluir Usuario</h4>
                </div>
                <div class="modal-body">

                    <?php
                    #inclui a view alterar usuario
                    include 'excluirUsuario.php';
                    ?>

                </div>
            </div>
        </div>
    </div>
    <?php
}
?>

```

```
<?php
}
?>

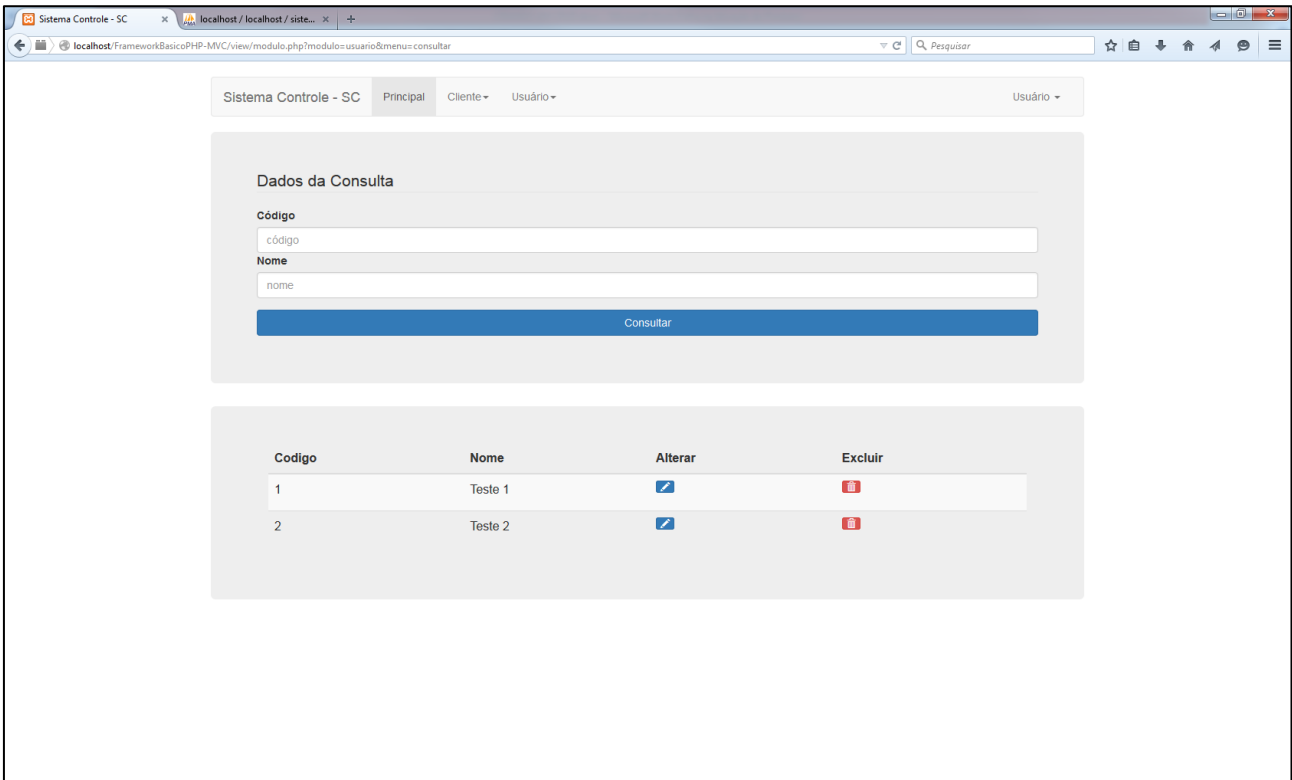
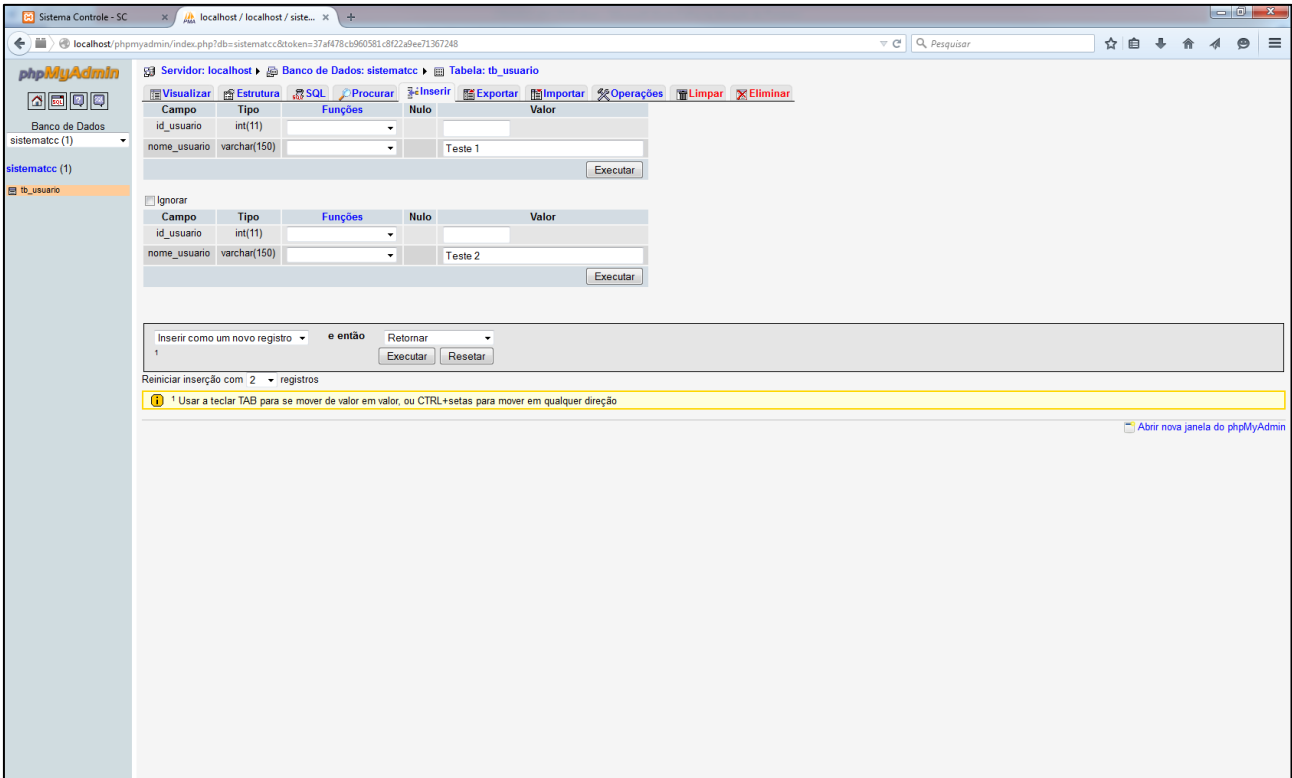
<!-- Script -->
<script>
$(document).ready(function() {
    $("#mytable #checkall").click(function() {
        if ($("#mytable #checkall").is(':checked')) {
            $("#mytable input[type=checkbox]").each(function() {
                $(this).prop("checked", true);
            });

        } else {
            $("#mytable input[type=checkbox]").each(function() {
                $(this).prop("checked", false);
            });
        }
    });

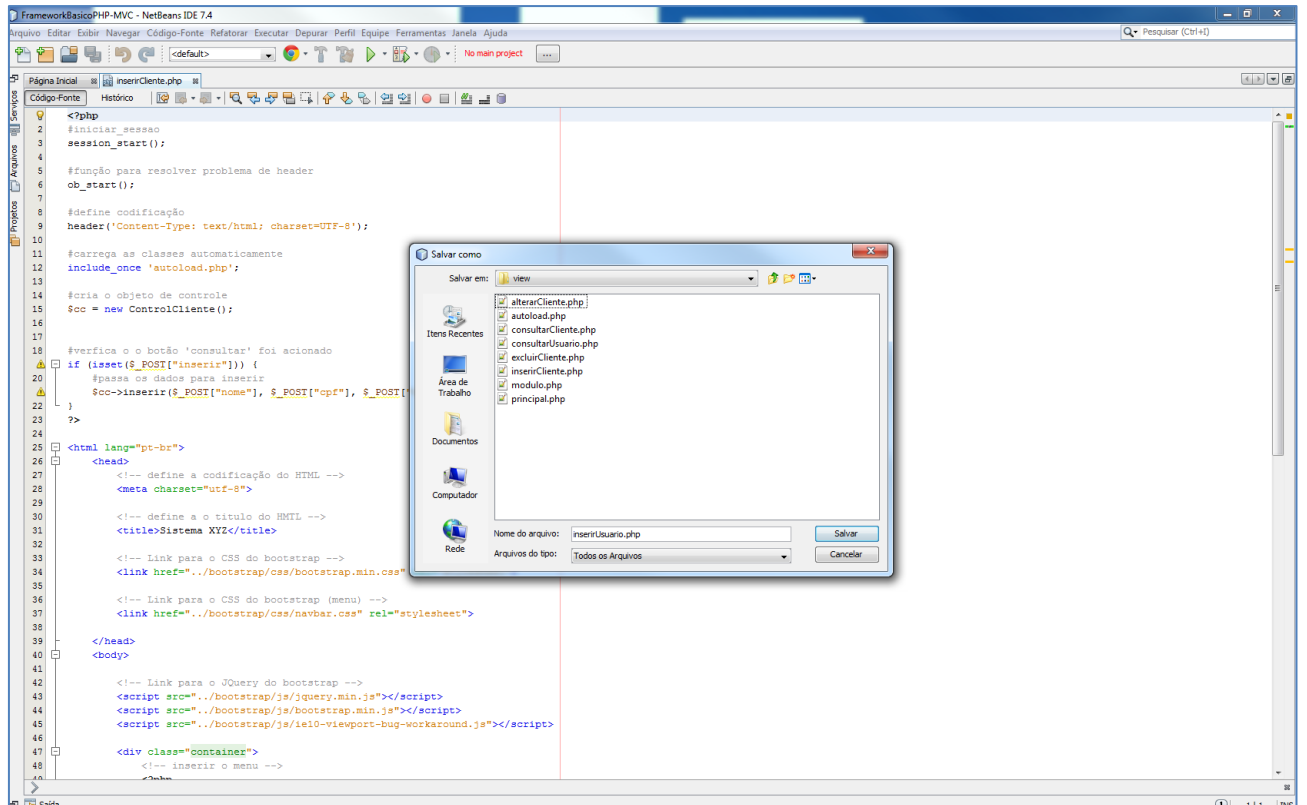
    $('[data-toggle=tooltip]').tooltip();
});
</script>

</body>
</html>
```

e) Popular o banco de dados manualmente, acessar o sistema e fazer o teste.

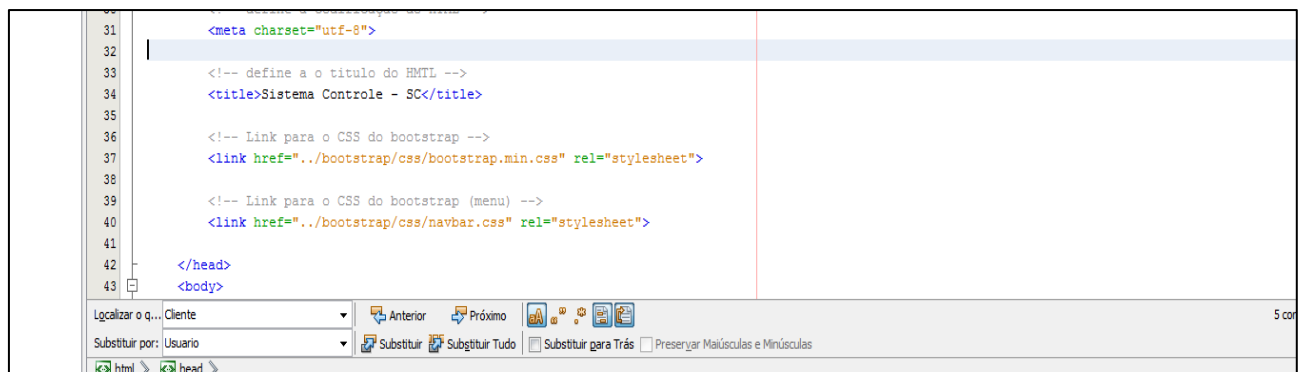


f) Criando a view Inserir Usuário. Abra uma view já existente *inserirCliente.php* e salve como *inserirUsuario.php*



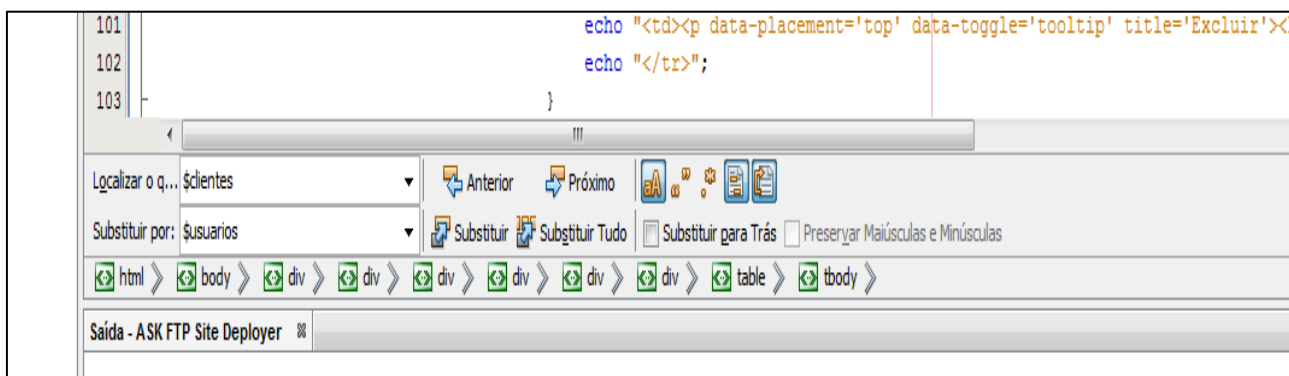
g) Substituição de nomes: No Netbeans acesse o menu Editar | Substituir...

Substituir: Cliente por Usuario



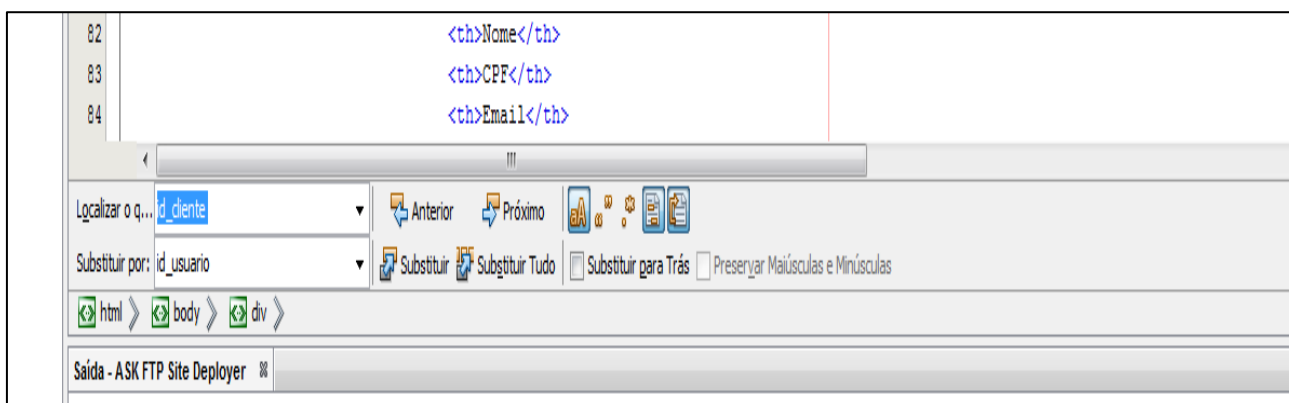
Clicar em substituir tudo

Substituir: \$clientes por \$usuarios



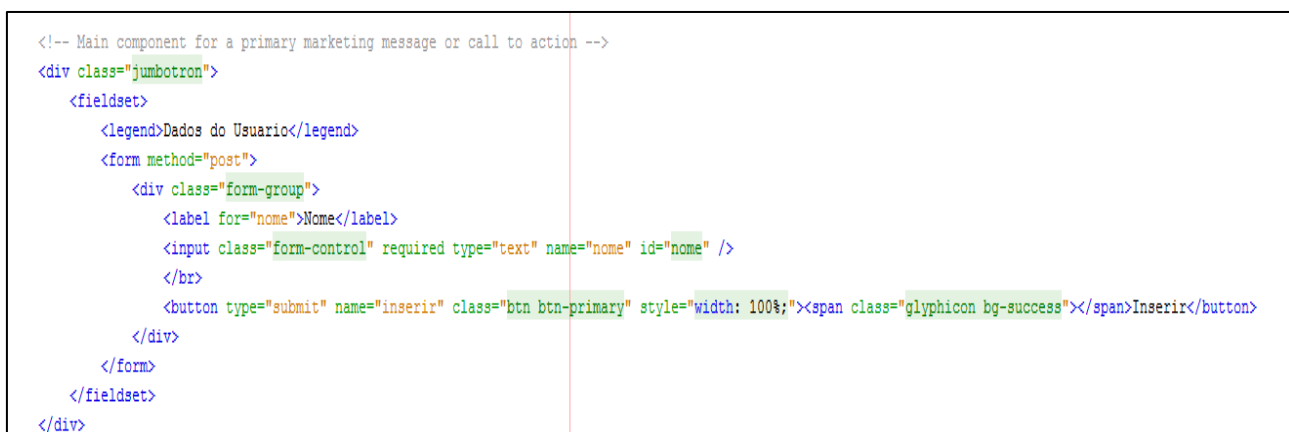
Clicar em substituir tudo

Substituir: id_cliente por id_usuario



Clicar em substituir tudo

h) Configurar o formulário de acordo com os campos.



i) Configurar os dados para inserção no banco de dados

```
4
5     #função para resolver problema de header
6     ob_start();
7
8     #define codificação
9     header('Content-Type: text/html; charset=UTF-8');
10
11     #carrega as classes automaticamente
12     include_once 'autoload.php';
13
14     #cria o objeto de controle
15     $cc = new ControlUsuario();
16
17
18     #verifica o o botão 'inserir' foi acionado
19     if (isset($_POST["inserir"])) {
20         #passa os dados para inserir
21         $cc->inserir($_POST["nome"]);
22     }
23     ?>
```

Código-fonte

```
<?php
#iniciar_sessao
session_start();

#função para resolver problema de header
ob_start();

#define codificação
header('Content-Type: text/html; charset=UTF-8');

#carrega as classes automaticamente
include_once 'autoload.php';

#cria o objeto de controle
$cc = new ControlUsuario();

#verifica o o botão 'inserir' foi acionado
if (isset($_POST["inserir"])) {
    #passa os dados para inserir
    $cc->inserir($_POST["nome"]);
}
?>

<html lang="pt-br">
<head>
    <!-- define a codificação do HTML -->
    <meta charset="utf-8">

    <!-- define a o titulo do HMTL -->
    <title>Sistema Controle - SC</title>

    <!-- Link para o CSS do bootstrap -->
    <link href="../bootstrap/css/bootstrap.min.css" rel="stylesheet">
```

```

<!-- Link para o CSS do bootstrap (menu) -->
<link href="../../bootstrap/css/navbar.css" rel="stylesheet">

</head>
<body>

    <!-- Link para o JQuery do bootstrap -->
    <script src="../../bootstrap/js/jquery.min.js"></script>
    <script src="../../bootstrap/js/bootstrap.min.js"></script>
    <script src="../../bootstrap/js/ie10-viewport-bug-workaround.js"></script>

    <div class="container">
        <!-- inserir o menu -->
        <?php
            #mostrar o menu
            $cc->menu();
            # $cc->alerta($_SESSION['msg']);
            ?>

        <!-- Main component for a primary marketing message or call to action -->
        <div class="jumbotron">
            <fieldset>
                <legend>Dados do Usuario</legend>
                <form method="post">
                    <div class="form-group">
                        <label for="nome">Nome</label>
                        <input class="form-control" required type="text" name="nome" id="nome" />
                        <br>
                        <button type="submit" name="inserir" class="btn btn-primary" style="width:
100%;"><span class="glyphicon bg-success"></span>Inserir</button>
                    </div>
                </form>
            </fieldset>
        </div>
    </body>
</html>

```

i) Acessar o sistema

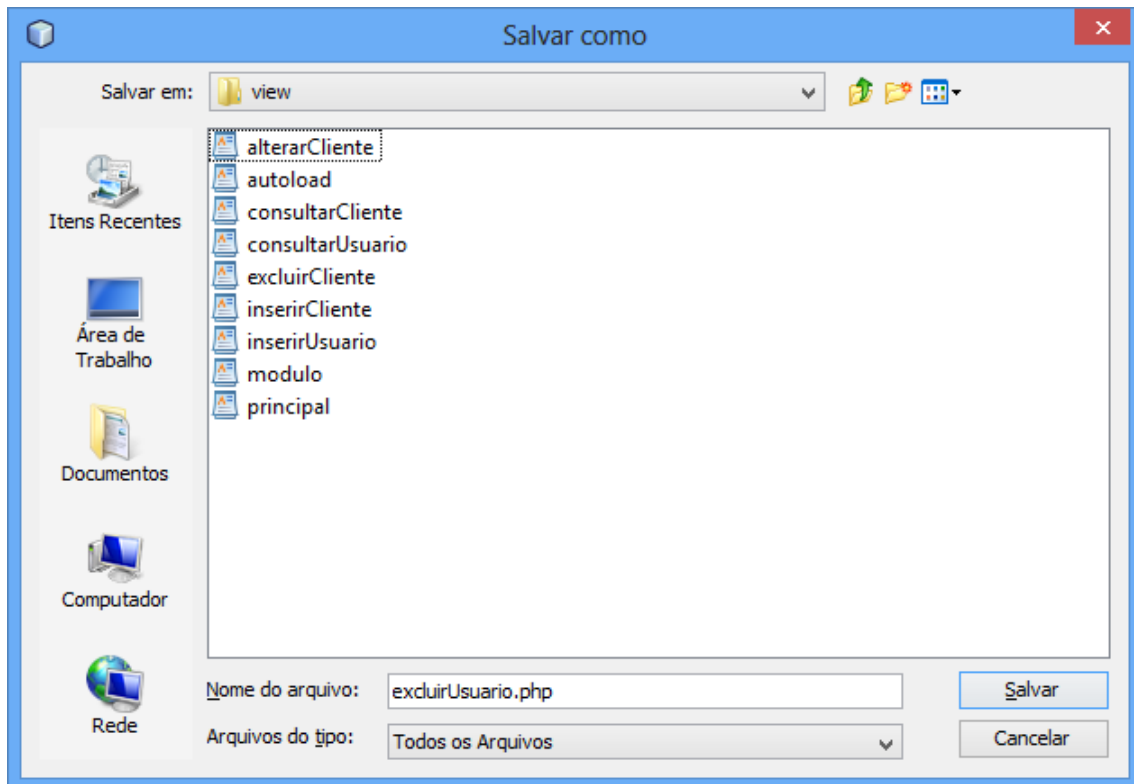
<http://localhost/FrameworkBasicoPHP-MVC/view/modulo.php?modulo=usuario&menu=inserrir>

The screenshot shows a web browser window with the address bar displaying `localhost/FrameworkBasicoPHP-MVC/view/modulo.php?modulo=usuario&menu=inserrir`. The page has a header with 'Sistema Controle - SC' and navigation tabs: 'Principal', 'Cliente', and 'Usuário'. A dropdown menu 'Usuário' is open. The main content area is titled 'Dados do Usuario' and contains a form with a label 'Nome' and a text input field containing 'Teste 4'. Below the input field is a blue button labeled 'Inserrir'.

The screenshot shows the same web browser window, but the address bar now displays `localhost/FrameworkBasicoPHP-MVC/view/modulo.php?modulo=usuario&menu=consultar`. The page layout is identical, but the main content area is titled 'Dados da Consulta'. It features a success message: 'Informação: Inserido com sucesso!'. Below this, there is a form with two input fields: 'Código' (containing 'código') and 'Nome' (containing 'nome'). A blue button labeled 'Consultar' is positioned below these fields. At the bottom of the page, there is a table displaying a list of users.

Codigo	Nome	Alterar	Excluir
1	Teste 1		
2	Teste 2		
3	Teste 3		
4	Teste 4		

j) Criando a view **excluirUsuario.php**. Abra uma view já existente *excluirCliente.php* e salve como *excluirUsuario.php*



Substituir:

\$clientes por \$usuarios

```
<?php
#invoca o método consultar do controle e passa como parâmetro o id do cliente
$usuarios_excluir = $cc->consultar($item[id_usuario], null);

#verificar se o botão "excluir" foi acionado
if (isset($_POST["excluir"])) {
    #passa o id do cliente para o controle realizar a exclusão
    $cc->excluir($_POST["id_usuario"]);
}

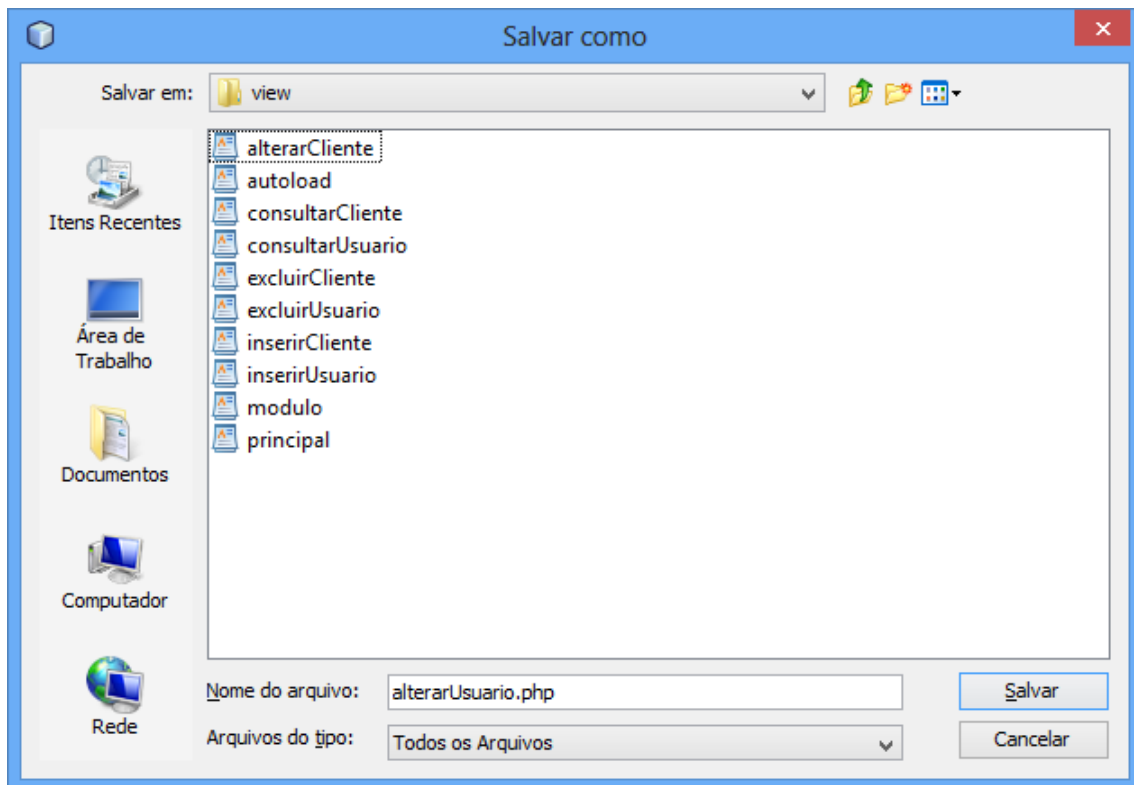
#mostrar os dados do cliente
foreach ($usuarios_excluir as $item_excluir) {
    ?>
    <fieldset>
        <form id="cliente" name="cliente" method="post" action="">
            <!-- dados do cliente -->
            <label for="id">Código</label>
            <input class="form-control" name="id_usuario" type="text" readonly="true" id="id"
value="<?php echo $item_excluir[id_usuario]; ?>" />
            <label for="nome">Nome</label>
            </br>
            <!-- botao para submeter o formulário -->
```

```

        <button id="enviar" type="submit" name="excluir" class="btn btn-danger btn-lg"
style="width: 100%;"><span class="glyphicon glyphicon-ok-sign"></span>Excluir</button>
    </form>
</fieldset>
<?php } ?>

```

I) Criando a view **alterarUsuario.php**. Abra uma view já existente *alteraCliente.php* e salve como *alterarUsuario.php*



```

<?php
#método para selecionar o cliente desejado
$usuarios_alterar = $cc->consultar($item[id_usuario], null);

#verificar se o botão "alterar" foi acionado
if (isset($_POST["alterar"])) {
    #passa os novos dados do cliente para o controle realizar a alteração
    $cc->alterar($_POST["id_usuario"], $_POST["nome_usuario"]);
}

#mostrar os dados do cliente
foreach ($usuarios_alterar as $item_alterar) {
    ?>
    <fieldset>
        <form method="post" action="">
            <!-- dados do cliente -->
            <label for="id">Código</label>

```

```
<input class="form-control" name="id_usuario" type="text" readonly="true"
id="id_usuario" value="<?php echo $item_alterar[id_usuario]; ?>" />
<label for="nome">Nome</label>
<input class="form-control" required type="text" name="nome_usuario"
id="nome_usuario" value="<?php echo $item_alterar[nome_usuario]; ?>" />
</br>
<!-- botao para submeter o formulário -->
<button type="submit" name="alterar" class="btn btn-warning btn-lg" style="width:
100%;"><span class="glyphicon glyphicon-ok-sign"></span>Alterar</button>
</form>
</fieldset>
<?php
}
?>
```