AGENDA10

PADRÃO MVC: MODEL VIEW CONTROLLER

MERGULHANDO NO TEMA...

MVC o que é?

MVC é a sigla para as palavras Model-View-Controller(Modelo-Visão-Controle), trata-se de um padrão de arquitetura para o desenvolvimento de software, podendo ser utilizado em desevenvolvimento WEB, Desktop ou Mobile,sem nenhuma objeção.

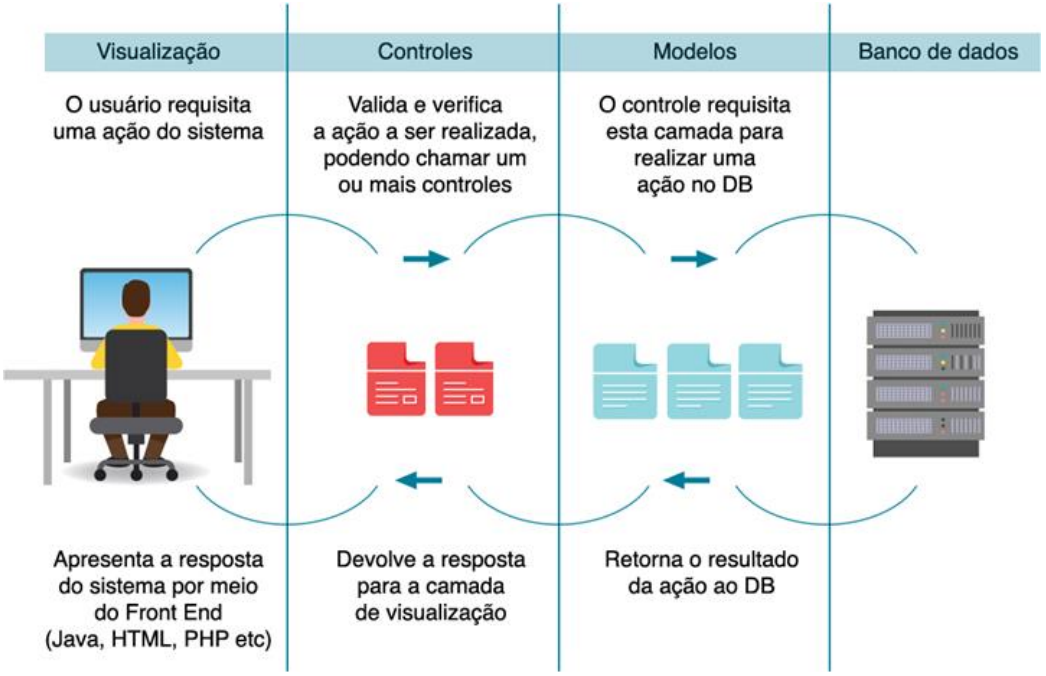


Imagem 02: Quadro explicativo do Padrão MVC.

Basicamente esse padrão tem como característica a separação de regras e lógicas do negócio, da apresentação ao usuário, o que permite ao desenvolvedor o maior controle sobre o projeto, tornando possível a manutenção individual de cada camada (MVC), proporcionando maior segurança no ciclo de desenvolvimento do projeto.

Como surgiu?

Christopher Alexander, um engenheiro civil da década de 70, criou o que é considerado o primeiro padrão de projeto. Por meio dessa iniciativa e conceitos apresentados, o desenvolvimento de software começou a ser encarado de uma maneira diferente e as primeiras documentações de padrões começaram a se desenvolver, para então Trygve Reenskaug, em 1979, começar a idealização do que hoje é chamado de padrão MVC.

Basicamente, Reenskaug gerou um padrão de arquitetura que projetava o softwareem três camadas independentes, essas camadas são: modelo, visão e controle (MVC), conforme pode ser observado na Imagem 02.

Quais Benefícios de usar um padrão / arquitetura de projeto?

Em qualquer área é essencial seguir padrões para atingir um nível mais elevado de profissionalismo. Podemos levantar alguns benefícios que serão agregados ao projeto ao utilizar padrões para o desenvolvimento. São eles:

- Aumento de produtividade;
- Uniformidade na estrutura do site;
- Redução de complexidade no código;
- Sites ficam mais fáceis de manter;
- Códigos e sites mais fáceis para manutenções e reuso;
- Facilita a documentação;
- Estabelece um vocabulário comum de projeto entre desenvolvedores;
- Permite a reutilização de módulos do site em outros sites;
- É considerada uma boa prática utilizar um conjunto de padrões para resolver problemas maiores que, sozinhos, não conseguiriam;
- Ajuda a construir sites confiáveis com arquiteturas testadas;
- Reduz o tempo de desenvolvimento de um projeto / site.

O padrão MVC

Quando utilizamos o padrão MVC, buscamos sua principal característica: isolar as regras de negócios, da lógica e da apresentação de interface ao usuário, buscando como principal benefício a possibilidade de que cada item, em cada camada, seja modificados sem a necessidade da alteração de outras camadas, tornando o projeto mais flexível e com uma grande possibilidade de reaproveitamento de código.

Vamos agora entender melhor essastrês camadas:

- Model;
- View;
- Controller.

Model

Basicamente é responsável pela modelagem e manipulação de informações de forma detalhada, então, todas as entidades / classes, consultas, cálculos e regras de negócio do site ou sistema devem permanecer nessa camada que tem acesso a toda e qualquer informação modelada ou não, proveniente,na maioria dos casos,de um Banco de Dados ou arquivo XML.

Exemplo:

- Modela os dados e o comportamento por atrás do processo de negócios;
- Preocupa-se com o armazenamento, manipulação e geração de dados;
- É um encapsulamento de dados e de comportamento independente da apresentação.

View

Como o próprio nome desta camada indica, fica a seu cargo apresentar tudo ao usuário final, então, é fácil afirmar que toda a interface faz parte dessa camada. Logo, todos os dados, informações, gráficos etc quando exibidos para o usuário final,é de responsabilidade desta camada.

Exemplo:

- Os elementos de exibição ao cliente: HTML, XML , ASP , Applets;
- Interface com o usuário;
- Camada usada para receber a entrada de dados e apresentar o resultado.

Controller

Responsável pelo fluxo de informação que passa pelo site. Esta camada irá gerir e definir quais dados/ informações ou regras devem ser acionadas e para onde serão encaminhadas para posteriormente serem exibidas.

Em resumo, esta camada age como uma intermediária, buscando as informações da camada Model e entregando para a camada View para a exibição ao usuário final. Portanto, o controle e as decisões do site devem fazer parte desta camada.

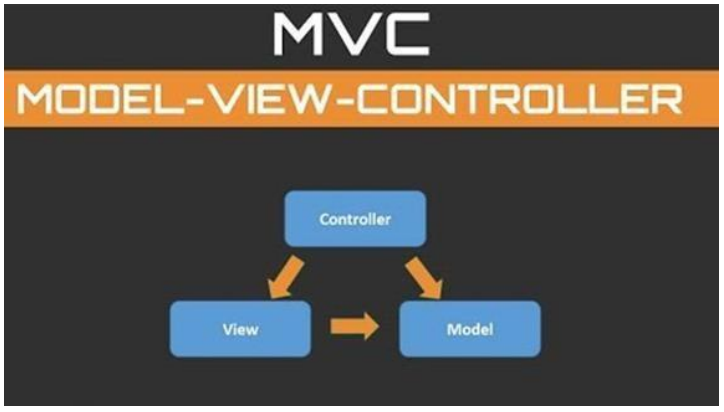
Exemplo:

- Um arquivo que recebe os dados de um formulário e os encaminha para a gravação dos mesmos em banco de dados ou encaminha o resultado de uma pesquisa para o usuário posteriormente visualizar.

Mas como funciona este padrão?
Assista aos vídeos a seguir e entenda o funcionamento deste padrão.

O que é MVC? Aprenda MVC de maneira fácil

Disponível em https://www.youtube.com/watch?v=ZW2JLtX4Dag.
Acessado em 02/05/2019.



Para entender melhor o funcionamento do modelo, imaginemos uma conversa entre o funcionamento do padrão e a relação entre suas camadas, (Tableless.com.br).

Imagem 03: : GEEaD - Diálogo entre as camadas MVC. Adaptado de Tableless.com.br[1]

VOCÊ NO COMANDO

Utilizando o que foi visto até agora....

1. Vamos relacionar alguns elementos de um site a uma camada do padrão MVC.

.PHP

A) Classe Usuário.

Essa classe traz todas as informações pertinentes, inclusive a persistência dessas informações em Banco de Dados.

CAMADA: _____

.CSS .HTML

B) Formulário de Contato.

Esse formulário traz os campos de texto para os usuários digitarem informações, e exibe informações sobre a empresa como e-mail, telefone etc.

CAMADA: _____

.PHP

C) Instância do Usuario.

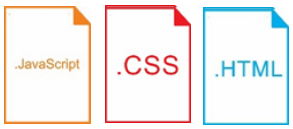
Esse arquivo instância a classe usuário, obtém as informações de um formulário e insere essas informações no objeto instânciado.

CAMADA: _____

[1] *Tableless.com.br – disponível em https://tableless.com.br/mvc-afinal-e-o-que/ .Acesso em 31/08/2018.*

D) SlideSHow de fotos.
Apresenta fotos de setores da empresa ao visitante do site.
CAMADA: _____

E) Classe Fornecedor.
Essa classe traz todas as informações pertinentes a fornecedores, inclusive a persistência dessas informações em Banco de Dados.
CAMADA: _____



F) Gráficos de Vendas.
Apresentao gráfico de venda dos últimos anos de empresa.
CAMADA: _____

G) Página Sobre a Empresa.
Esta página traz todas as informações da empresa, desde seu surgimento até o momento atual.
CAMADA: _____

H) Validação de Fornecedor.
Código que permite a liberação do pagamento, somente após a entrega do pedido com as métricas definidas no contratado pelo fornecedor.
CAMADA: _____

I) Data e hora do site.
Apresenta data e hora para os usuários que acessam a página.
CAMADA: _____

A seguir, confira se você conseguiu resolver os desafios propostos!

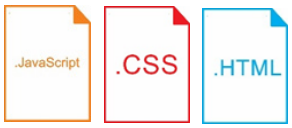
A) Classe Usuário.
CAMADA: MODEL

B) Formulário de Contato.
CAMADA: VIEW

C) Instância do Usuario.
CAMADA: CONTROLLER

D) Slide Show de fotos.
CAMADA: VIEW

E) Classe Fornecedor.
CAMADA: MODEL



F) Gráficos de Vendas.
CAMADA: VIEW

G) Página Sobre a Empresa.
CAMADA: VIEW

H) Validação de Fornecedor.
CAMADA: MODEL

I) Data e hora do site.
CAMADA: VIEW

Explicações:

CAMADA MODEL: Como descrevemos anteriormente, esta camada é, basicamente responsável pela modelagem e manipulação de informações de forma detalhada, então, todas as entidades/classes, consultas, cálculos e regras de negócio do site ou sistema devem permanecer nessa camada.
Então o resultado deve ficar:
Classe Fornecedor:
Classe Usuário. Validação de Fornecedor.

CAMADA VIEW: responsável por apresentar tudo para que o usuário final visualize e interaja com a interface então, é fácil afirmar que toda a interface faz parte dessa camada, logo, fazem parte desta camada todos os dados, como informações, gráficos etc.
Então nessa camada deve ficar:
Formulário de Contato.
Página Sobre a Empresa.
Data e hora do site.
Gráficos de Vendas.
SlideSHow de fotos.

CAMADA CONTROLLER: controla o fluxo de informação que passa pelo site. Esta camada irá gerir e definir quais dados/informações ou regras devem ser acionadas e para onde serão encaminhadas para posteriormente serem exibidas.
Então, nessa camada deve ficar:

AGENDA 11

PROJETO COMPLETO CAMADA MODEL

MERGULHANDO NO TEMA...

Com os conhecimentos adquiridos até agora sobre o Padrão MVC e PHP Orientado a Objetos, iniciaremos a construção do projeto completo. Nessa agenda, vamos estudar e aplicar os conhecimentos da camada Model do Padrão MVC.

Para aplicar o padrão MVC em um projeto com PHP Orientado a Objetos, precisamos modelar as classes e manipular os dados para deixar a camada Model pronta.

Imagem 01 - freepik.com

Agora é hora de colocar em prática seus conhecimentos em PHP Orientado a Objetos e Arquitetura MVC. Vamos começar pelo desenvolvimento da camada model do projeto! Vamos lá?

Banco de dados

Antes de mais nada, será necessário criar o Banco de Dados para o projeto. A seguir, é apresentado o diagrama do Banco de Dados com as tabelas Usuário e Formação Acadêmica. Atente-se às tabelas, aos atributos e ao relacionamento entre elas:

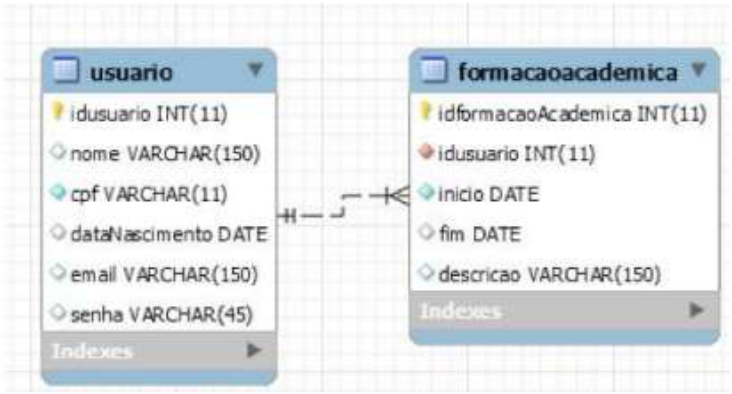


Imagem 02 - Diagrama Lógico de Banco de dados

Para este projeto, utilizamos o banco de dados MySql e a mesma base de dados que será utilizada na próxima agenda. Segue o script:

```
CREATE SCHEMA IF NOT EXISTS `projeto_final` DEFAULT CHARACTER SET latin1 ;
USE `projeto_final` ;
-- Table `projeto_final`.`usuario`
CREATE TABLE IF NOT EXISTS `projeto_final`.`usuario` (
  `idusuario` INT(11) NOT NULL AUTO_INCREMENT,
  `nome` VARCHAR(150) NULL DEFAULT NULL,
  `cpf` VARCHAR(11) NOT NULL,
  `dataNascimento` DATE NULL DEFAULT NULL,
  `email` VARCHAR(150) NULL DEFAULT NULL,
  `senha` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`idusuario`),
  UNIQUE INDEX `cpf_UNIQUE` (`cpf` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;
-- Table `projeto_final`.`formacaoAcademica`
CREATE TABLE IF NOT EXISTS `projeto_final`.`formacaoAcademica` (
  `idformacaoAcademica` INT(11) NOT NULL AUTO_INCREMENT,
  `idusuario` INT(11) NOT NULL,
  `inicio` DATE NOT NULL,
  `fim` DATE NULL DEFAULT NULL,
  `descricao` VARCHAR(150) NULL DEFAULT NULL,
  PRIMARY KEY (`idformacaoAcademica`),
  INDEX `IDUSUARIO_idx` (`idusuario` ASC),
  CONSTRAINT `IDUSUARIO`
    FOREIGN KEY (`idusuario`)
    REFERENCES `projeto_final`.`usuario` (`idusuario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = latin1;
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Camada Model

Para colocar em prática, precisamos criar uma pasta chamada Model, dentro da pasta raiz do servidor apache que você utiliza. Relembrando que dentro desta pasta deve ficar a modelagem e a manipulação de informações. Assim, todas as entidades/classes, as consultas, os cálculos e as regras de negócio do site ou sistema devem permanecer nessa camada.

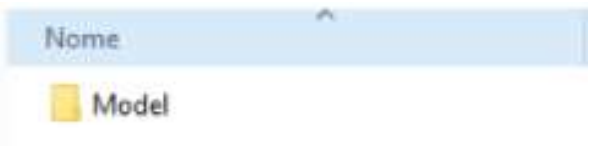


Imagem 03 - Pasta Model

Classe ConexaoBD

Neste momento, vamos modelar uma classe que será responsável por realizar a conexão com o banco de dados. Seu nome será “ConexaoBD”, então dentro da pasta Model, crie este arquivo PHP.

Não se esqueça de colocar os delimitadores PHP“<?php?>”. O desenvolvimento desta classe será baseado no diagrama a seguir.

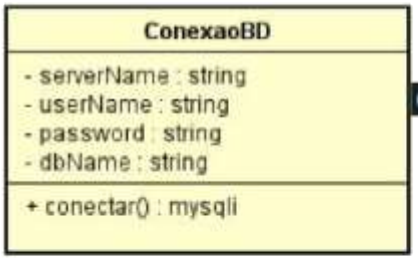


Imagem 04 - Diagrama da Classe ConexaoBD

Logo após os delimitadores, deve-se criar a classe com o seguinte código:

```
Class ConexaoBD{}
```

Dentro destas chaves, começaremos criando os 4 atributos privados:

- serverName - nome ou ip do servidor;
- userName - nome do usuário de conexão ao banco de dados;
- password - senha para conexão ao banco de dados;
- dbName - nome da base de dados deste projeto.

O código deve ficar dessa forma:

```
class ConexaoBD{

    private $serverName = "localhost";
    private $userName = "root";
    private $password = "usbw";
    private $dbName = "projeto final";
}
```

Observação: os valores atribuídos a cada atributo fazem referência ao banco de dados usado para desenvolver este exemplo. Portanto, você deve colocar os dados do seu banco de dados.

Agora basta desenvolver o método específico com o código a seguir para finalizar a primeira classe.

```
    public function conectar()
    {
        $conn = new mysqli($this->serverName, $this->userName, $this->password, $this->dbName);
        return $conn;
    }
```

IMPORTANTE: A função conectar() deve ser inserida dentro da classe ConexaoBD.

Este método sem parâmetros, quando invocado, cria uma conexão com o banco de dados de acordo com os valores de seus atributos e retorna essa conexão para quem o invocou.

Camada Usuário

Agora vamos modelar a classe Usuário que será responsável por gerenciar os dados do usuário no projeto. Seu nome será “Usuario”, então novamente dentro da pasta Model, crie este arquivo PHP, não se esquecendo de colocar os delimitadores php“”. O desenvolvimento desta classe será baseado no diagrama a seguir:

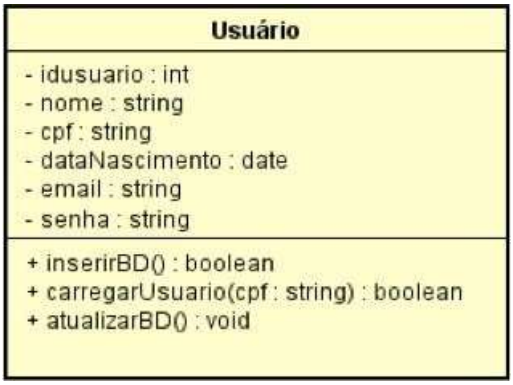


Imagem 05 – Diagrama da Classe Usuário

Logo após os delimitadores, devemos criar a classe e seus atributos:

- idusuário - código único de cada registro de usuário;
- nome - nome do usuário;
- cpf - CPF do usuário;
- dataNascimento - data de nascimento do usuário;
- email - e-mail do usuário;
- senha - senha para acesso ao site.

A codificação deverá ficar dessa forma:

```
class Usuario
{
    private $id;
    private $nome;
    private $cpf;
    private $email;
    private $dataNascimento;
    private $senha;
}
```

O próximo passo é criar todos os métodos getters e setters para acesso aos atributos privados da classe. Então, seguindo os padrões de projeto, vamos codificá-los:

```php
//ID
    public function setID($id)
    {
        $this->id = $id;
    }
    public function getID()
    {
        return $this->id;
    }

    //nome
    public function setNome($nome)
    {
        $this->nome = $nome;
    }
    public function getNome()
    {
        return $this->nome;
    }

    //cpf
    public function setCPF($cpf)
    {
        $this->cpf = $cpf;
    }
       public function getCPF()
    {
        return $this->cpf;
    }
```

```php
 //Email
public function setEmail($email)
{
    $this->email = $email;
}
public function getEmail()
{
    return $this->email;
}

//Data de nascimento
public function setDataNascimento($dataNascimento)
{
    $this->dataNascimento = $dataNascimento;
}
public function getDataNascimento()
{
    return $this->dataNascimento;
}

// Senha
public function setSenha($senha)
{
    $this->senha = $senha;
}
public function getSenha()
{
    return $this->senha;
}
```

Agora vamos partir para a codificação dos métodos específicos, que são três: inserirBD, carregarUsuário e atualizarBD. Todos estão relacionados a operações com o Banco de Dados de devem ser inseridos na classe Usuário.

Método inserirBD

Primeiramente vamos codificar o método “inserirBD”, dentro da classe Usuário, sua função será inserir no Banco de Dados as informações do usuário quando for invocado, mas neste primeiro momento não vamos inserir todas as informações. Ao realizar o primeiro contato com a página e, consequentemente, realizar o cadastro, o usuário precisará apenas fornecer: Nome, CPF, e-mail e senha. Esse método é bem popular para agilizar o cadastro e o restante das informações podem ser inseridas em futuras atualizações.

Vamos codificar?

```
    public function inserirBD()
    {
        require once 'ConexaoBD.php';

        $con = new ConexaoBD();
        $conn = $con->conectar();
        if ($conn->connect error) {
            die("Connection failed: " . $conn->connect error);
        }
        $sql = "INSERT INTO usuario (nome, cpf, email, senha)
        VALUES ('".$this->nome."', '".$this->cpf."', '".$this->email."','".$this->senha."')";

        if ($conn->query($sql) === TRUE) {
            $this->id = mysqli insert id($conn);
            $conn->close();
            return TRUE;

        } else {
            $conn->close();
            return FALSE;
        }
    }
}
```

Perceba que o método foi definido como público para ser possível sua utilização após a instância de um objeto da classe Usuário; depois, é necessária a inclusão da classe conexãoBD, para que seja possível a utilização da mesma.

Então, na sequência devemos:

Logo após os delimitadores, devemos criar a classe e seus atributos:

- idusuário - código único de cada registro de usuário;
- nome - nome do usuário;
- cpf - CPF do usuário;
- dataNascimento - data de nascimento do usuário;
- email - e-mail do usuário;
- senha - senha para acesso ao site.

Se tudo ocorrer corretamente, sabemos que a conexão foi realizada e assim devemos confeccionar a sentença SQL:

```
$sql = "INSERT INTO usuario (nome, cpf, email, senha)
            VALUES ('".$this->nome."', '".$this->cpf."', '".$this->email."','".$this->senha."')";
```

Observação: Perceba que utilizamos os atributos da classe para montar essa sentença, ou seja, o programador será forçado a popular os atributos por meio dos métodos getters and setters para que o desenvolvimento fique padronizado.
Por fim, executamos a sentença SQL e verificamos se tudo ocorreu bem:

if ($conn->query($sql) === TRUE).

- Caso positivo, obtemos o id gerado no banco de dados para que seja inserido no objeto, fechamos a conexão e retornamos TRUE.
- Caso negativo, fechamos a conexão e retornamos FALSE.

Pronto, o primeiro método específico foi finalizado!
Vamos para o próximo método: carregarUsuario.

Método carregarUsuario

```php
public function carregarUsuario($cpf)
    {
        require_once 'ConexaoBD.php';

        $con = new ConexaoBD();
        $conn = $con->conectar();
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }
            $sql = "SELECT * FROM usuario WHERE cpf =  ".$cpf ;
            $re = $conn->query($sql);
            $r = $re->fetch_object();
            if($r != null)
            {
                $this->id = $r->idusuario;
                $this->nome = $r->nome;
                $this->email = $r->email;
                $this->cpf = $r->cpf;
                $this->dataNascimento = $r->dataNascimento;
                $this->senha = $r->senha;
                $conn->close();
                return true;
            }
            else
            {
                $conn->close();
                return false;
            }
    }
```

Este método segue o mesmo padrão:

- Inclusão da Classe ConexaoBD;
- Instância do Objeto da Classe ConexãoBD;
- Conexão ao Banco de Dados, com verificação do sucesso ou não;
- Confecção da sentença SQL;
- Execução da sentença, com verificação do sucesso ou não.

A diferença é que em caso positivo, deve-se popular os dados dos objetos com o resultado da consulta ao Banco de Dados, fechando, em seguida, a conexão e por fim retornando TRUE.

```
            $this->id = $r->idusuario;
            $this->nome = $r->nome;
            $this->email = $r->email;
            $this->cpf = $r->cpf;
            $this->dataNascimento = $r->dataNascimento;
            $this->senha = $r->senha;
            $conn->close();
            return true;
```

Agora vamos ao último método da classe Usuário!

Método atualizarBD

Concluindo, o último método específico da classe Usuário: atualizarBD, terá como função atualizar os dados do usuário no Banco de Dados.

```
public function atualizarBD()
    {
        require_once 'ConexaoBD.php';
        $con = new ConexaoBD();
        $conn = $con->conectar();
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }

        $sql = "UPDATE usuario SET nome = '".$this->nome."', cpf = '". $this->cpf."', dataNascimento =
'". $this->dataNascimento."',
        email='".$this->email."'  WHERE idusuario ='". $this->id. "'"    ;
        if ($conn->query($sql) === TRUE) {
            $conn->close();
            return TRUE;

        } else {
            $conn->close();
            return FALSE;
        }
    }
```

Seguindo o mesmo padrão:

- Inclusão da classe ConexaoBD;
- Instância do objeto da Classe ConexãoBD;
- Conexão ao Banco de Dados, com verificação do sucesso ou não;
- Confecção da sentença SQL;
- Execução da sentença com verificação do sucesso ou não.

A diferença é que em caso positivo, atualizam se os dados do Banco com os dados populados no objeto instanciado, em seguinda, fecha se a conexão, retornando a TRUE. Pronto! Está finalizada a classe Usuário.

Fim da classe Usuário!

Classe FormacaoAcad

Agora vamos modelar a classe FormacaoAcad, que será responsável por gerenciar os dados da formação acadêmica do usuário. Crie o arquivo PHP com o nome "FormacaoAcad", novamente dentro da pasta Model.

Não se esqueça de colocar os delimitadores php "<?php?>". O desenvolvimento desta classe será baseado no diagrama a seguir:

Logo após os delimitadores, devemos criar a classe e seus atributos, sabendo que:

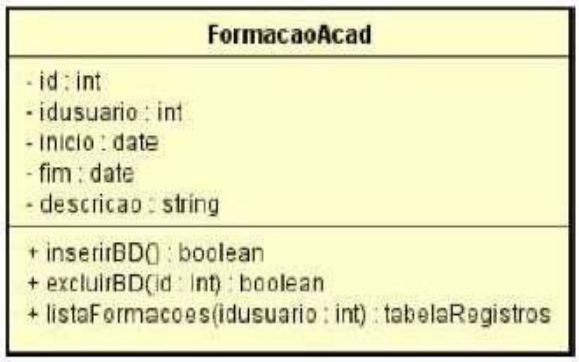


Imagem 6 - Diagrama da Classe FormacaoAcad

- id é o código único de cada registro de formação acadêmica;
- idusuario é o código do usário a quem pertence essa formação;
- início é a data do início da formação;
- fim é a data do fim da formação;
- descrição é a descrição da formação acadêmica.

Como resultado, devemos obter o seguinte código:

```php
class FormacaoAcad
{
    private $id;
    private $idusuario;
    private $inicio;
    private $fim;
    private $descricao;
}
```

Assim como na classe Usuário, o próximo passo é criar todos os métodos getters e setters de cada um desses atributos. Eles darão acesso aos atributos privados desta classe. Seguindo os padrões de projeto, codifica-se dessa forma:

```
        //ID
public function setID($id)
{
    $this->id = $id;
}
public function getID()
{
    return $this->id;
}
//idusuario
public function setIdUsuario($idusuario)
{
    $this->idusuario = $idusuario;
}
public function getIdUsuario()
{
    return $this->idusuario;
}

//inicio
public function setInicio($inicio)
{
    $this->inicio = $inicio;
}
   public function getInicio()
{
    return $this->inicio;
}

 //fim
public function setFim($fim)
{
    $this->fim = $fim;
}
public function getFim()
{
    return $this->fim;
}

//Descrição
public function setDescricao($descricao)
{
    $this->descricao = $descricao;
}
public function getDescricao()
{
    return $this->descricao;
}
```

formacaoAcademica

Agora vamos partir para a codificação dos métodos específicos, que são três: inserirBD, excluirBD e listaFormacoes. Todos estão relacionados a operações com o Banco de Dados. Para a criação desses métodos, siga o exemplo da codificação dos métodos da classe Usuário. O primeiro método, "inserirBD", é exatamente como o desenvolvido para a classe Usuário, apenas mudando para a tabela (INSERT INTO). Para codificar, basta seguir o mesmo padrão. Veja:

```php
public function inserirBD()
    {
        require_once 'ConexaoBD.php';

        $con = new ConexaoBD();
        $conn = $con->conectar();
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }

       $sql = "INSERT INTO formacaoAcademica (idusuario, inicio, fim, descricao)
        VALUES ('".$this->idusuario."','".$this->inicio."','".$this->fim."','".$this->descricao."')";

         if ($conn->query($sql) === true) {
            $this->id = mysqli_insert_id($conn);
            $conn->close();
            return true;

        } else {
            $conn->close();
            return false;
        }
    }
```

Vamos para o próximo método: excluirBD.

```php
public function excluirBD($id)
    {
        require_once 'ConexaoBD.php';
                $con = new ConexaoBD();
        $conn = $con->conectar();
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }

       $sql = "DELETE FROM formacaoAcademica WHERE idformacaoAcademica = '".$id ."';";

        if ($conn->query($sql) === true) {

            $conn->close();
            return true;

        } else {
            $conn->close();
            return false;
        }
    }
```

Este método segue o mesmo padrão:

- Inclusão da classe ConexaoBD;
- Instância do objeto da Classe ConexãoBD;
- Conexão ao Banco de Dados, com verificação do sucesso ou não;
- Confecção da sentença SQL;
- Execução da sentença com verificação do sucesso ou não.

A diferença é que precisa de um id (parâmetro) para realizar a exclusão do registro correto. Então, em caso positivo, será excluído o registro do Banco de Dados e em seguida será fechada a conexão, retornando a TRUE. Caso negativo, fecha se a conexão e retorna a FALSE.

Por fim, codifique o método listaFormacoes:

```php
public function listaFormacoes($idusuario)
    {
        require_once 'ConexaoBD.php';

        $con = new ConexaoBD();
        $conn = $con->conectar();
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }

        $sql = "SELECT * FROM formacaoAcademica WHERE idusuario =  '".$idusuario."'" ;
        $re = $conn->query($sql);
        $conn->close();
        return $re;
    }
```

Este método segue inicialmente o mesmo Padrão:

- Inclusão da classe ConexaoBD;
- Instância do objeto da Classe ConexãoBD;
- Conexão ao Banco de Dados, com verificaração do sucesso ou não;
- Confecção da sentença SQL;

Porém, além de receber o id do usuário por meio de passagem de parâmetro, esse método retorna um ou mais registros como resultado da consulta do Banco de Dados.

```
$re = $conn->query($sql);
$conn->close();
return $re;
```

VOCÊ NO COMANDO

Analise o diagrama lógico de Banco de Dados, nomeado por ExperienciaProfissional (Imagem 07):

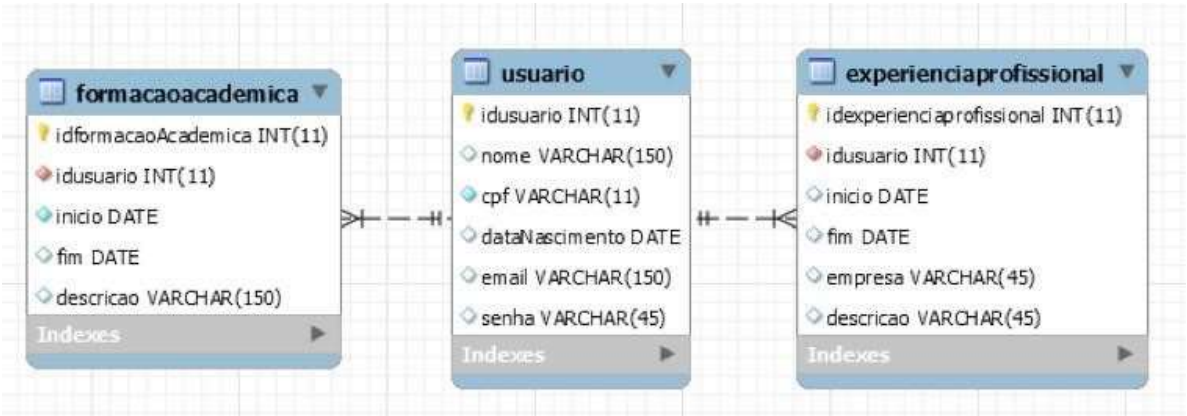


Imagem 7 - Diagrama Lógico de Banco de dados

Analise também o diagrama de classe que foi gerado para a classe ExperienciaProfissional (Imagem 08).

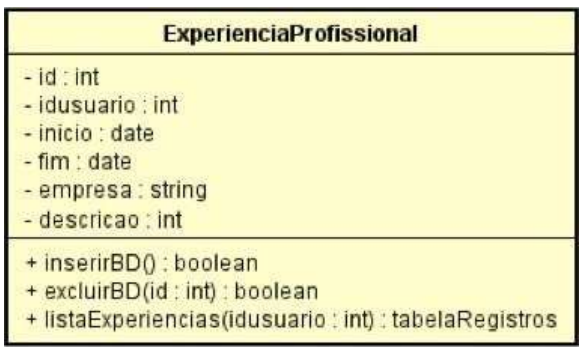


Imagem 8 - Diagrama de classe Experiência Profissional

Tendo as imagens como referência, faça o que se pede:

1. **Crie a Classe ExperienciaProfissional dentro da Pasta Model, de acordo com o diagrama acima.**
2. **Gere os Atributos de acordo com as seguintes informações:**
   a. id- código único da experiência cadastrada;
   b. idusuario- código único do usuário;
   c. inicio- início da experiência profissional;
   d. fim- final da experiência profissional;
   e. empresa- local onde obtida a experiência profissional;
   f. descricao- explicação do cargo e aprendizados.
3. **Crie os métodos específicos:**
   a. inserirBD;
   b. excluirBD;
   c. lista.

Dica: Utilize a classe formação acadêmica como base para o desenvolvimento.
Caso esteja com dificuldade, segue script SQL para criação da tabela ExperienciaProfissional:

```
CREATE TABLE `projeto_final`.`experienciaprofissional` (
  `idexperienciaprofissional` INT NOT NULL AUTO_INCREMENT,
  `idusuario` INT NOT NULL,
  `inicio` DATE NULL,
  `fim` DATE NULL,
  `empresa` VARCHAR(45) NULL,
  `descricao` VARCHAR(45) NULL,
  PRIMARY KEY (`idexperienciaprofissional`),
  INDEX `idUser_idx` (`idusuario` ASC),
  CONSTRAINT `idUser`
    FOREIGN KEY (`idusuario`)
    REFERENCES `projeto_final`.`usuario` (`idusuario`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

A seguir, confira se você conseguiu resolver o desafio proposto, mas saiba que a solução apresentada é apenas uma dentre algumas possibilidades e variações. Com o nome “ExperiênciaProfissional”, novamente dentro da pasta Model, crie o arquivo PHP. Não se esqueça de colocar os delimitadores php “<?php?>” .
O desenvolvimento dessa classe deve ser baseado no diagrama de Classe ExperienciaProfissional (Imagem 08).
A partir dos dados desse diagrama:
Crie a classe e seus atributos com seus respectivos métodos getters and setters:

```php
class ExperienciaProfissional
{
    private $id;
    private $idusuario;
    private $inicio;
    private $fim;
    private $empresa;
    private $descricao;


    //ID
    public function setID($id)
    {
        $this->id = $id;
    }
    public function getID()
    {
        return $this->id;
    }
    //idusuario
    public function setIdUsuario($idusuario)
    {
        $this->idusuario = $idusuario;
    }
    public function getIdUsuario()
    {
        return $this->idusuario;
    }
```

```php
public function setInicio($inicio)
{
    $this->inicio = $inicio;
}
   public function getInicio()
{
    return $this->inicio;
}

 //fim
public function setFim($fim)
{
    $this->fim = $fim;
}
public function getFim()
{
    return $this->fim;
}

//Empresa
public function setEmpresa($empresa)
{
    $this->empresa = $empresa;
}
public function getEmpresa()
{
    return $this->empresa;
}

//Descrição
public function setDescricao($descricao)
{
    $this->descricao = $descricao;
}
public function getDescricao()
{
    return $this->descricao;
}
```

Veja a construção do Método inserirBD:

```
public function inserirBD()
    {
        require_once 'ConexaoBD.php';

        $con = new ConexaoBD();
        $conn = $con->conectar();
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }

       $sql = "INSERT INTO experienciaprofissional (idusuario, inicio, fim, empresa, descricao)
        VALUES ('".$this->idusuario."','".$this->inicio."','".$this->fim."','".$this-
>empresa."','".$this->descricao."')";


        if ($conn->query($sql) === true) {
            $this->id = mysqli_insert_id($conn);
            $conn->close();
            return true;

        } else {
            $conn->close();
            return false;
        }
    }
```

Veja a construção do Método excluirBD:

```
public function excluirBD($id)
    {
        require_once 'ConexaoBD.php';

        $con = new ConexaoBD();
        $conn = $con->conectar();
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }

       $sql = "DELETE FROM experienciaprofissional WHERE idexperienciaprofissional = '".$id ."';";


        if ($conn->query($sql) === true) {

            $conn->close();
            return true;

        } else {
            $conn->close();
            return false;
        }
    }
```

Veja a construção do Método listarExperiencias:

```php
public function listaExperiencias($idusuario)
    {
        require_once 'ConexaoBD.php';

        $con = new ConexaoBD();
        $conn = $con->conectar();
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }

        $sql = "SELECT * FROM experienciaProfissional WHERE idusuario =  '".$idusuario."'" ;
        $re = $conn->query($sql);
        $conn->close();
        return $re;
    }
```

Agora que você finalizou a codificação, confira o resultado e converse com o seu professor-tutor sobre o sucesso do resultado das suas implementações, para esclarecer alguma dúvida ou ainda para solicitar ajuda!