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 02 - 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:

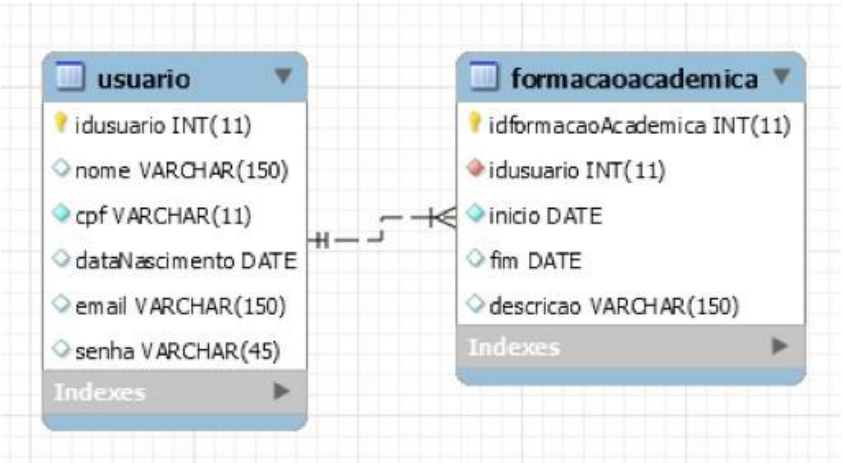


Imagem 03 - Diagrama Lógico de Banco de dados

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.

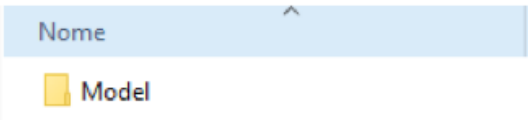


Imagem 04 - Pasta Model

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.

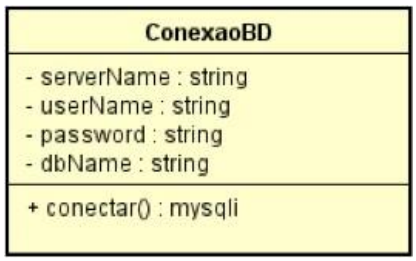


Imagem 05 - Diagrama da Classe ConexaoBD

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";
}
```

Obs.: 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;
    }
```

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.

Classe 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"<?php?>" . O desenvolvimento desta classe será baseado no diagrama a seguir:

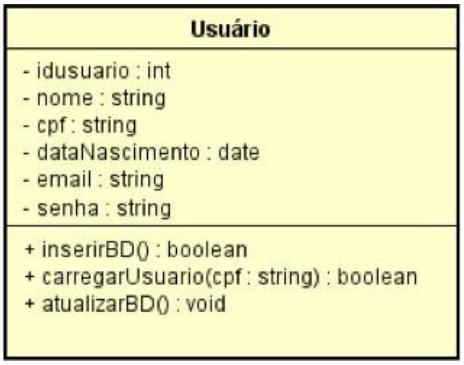


Imagem 05 - Diagrama da Classe Usuário

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:

```php
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;
    }

     //Email
    public function setEmail($email)
    {
        $this->email = $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 especícios, que são três: inserirBD, carregarUsuário e atualizarBD.

Todos estão relacionados a operações com o Banco de Dados.

Método inserirBD

Primeiramente vamos codificar o método “inserirBD”, 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 preci- sará 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:

- Instanciar objeto: $con = new ConexaoBD();
- Conectar ao Banco de dados e retornar essa conexão, atribuindo a um novo objeto: $conn = $con->conectar();
- Verificar se a conexão foi realizada: if ($conn->connect_error);

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."')";
```

Obs: 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

```
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;
- Confeção 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 resultado da consulta ao Banco de Dados, fechando, em seguida, a conexão e por fim retornando a 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;
        }
    }
```

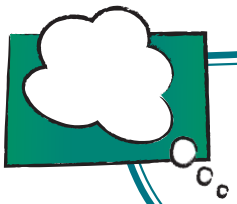
Seguindo o mesmo padrão:

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

Classe FormacaoAcad

Agora vamos modelar a classe FormacaoAcad que será responsável por gerenciar os dados da formação acadêmida do usuário. Cire 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:

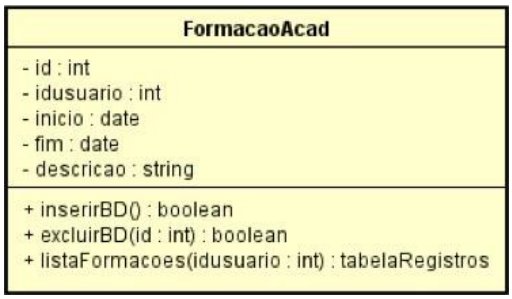


Imagem 6 - Diagrama da Classe FormacaoAcad

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

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

Como resultado, devemos obter o seguinte código:

```
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 lista-Formacoes. 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:

```
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.

```
   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;
- Confeção 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:

```
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;
- Confeção 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):

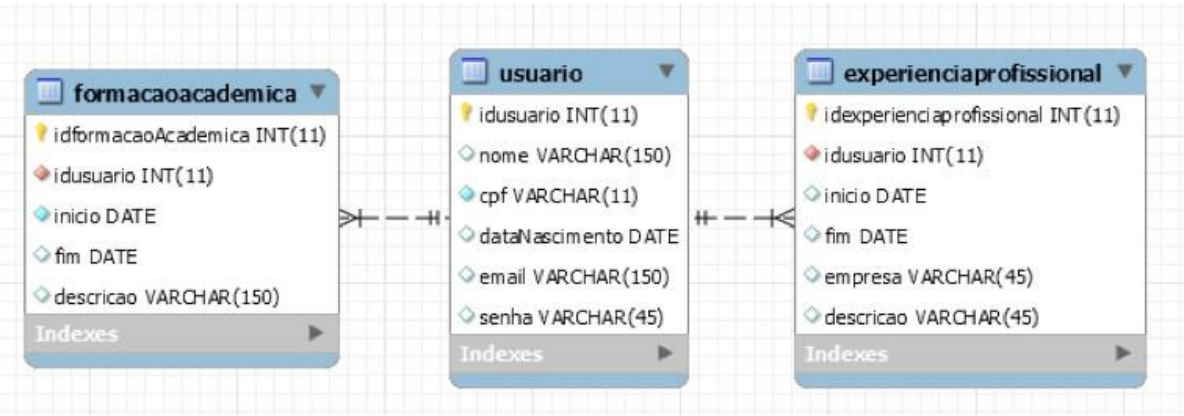


Imagem 7 - Diagrama Lógico de Banco de dados

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

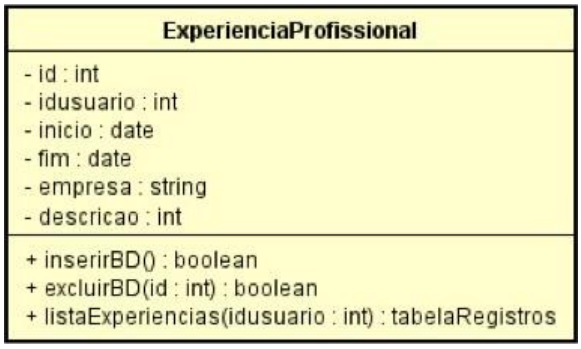


Imagem 8 - Diagrama de classe Experiência Profissional

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

1. Crie a Classe ExperiênciaProfissional 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. descrição – explicação do cargo e aprendizados.

3. Crie os método 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 de 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:

```
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;
    }

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

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 listarExperiências:

```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!