

# LÓGICA DE PROGRAMAÇÃO



Técnico Integrado em  
**INFORMÁTICA**

Professor Jefferson Chaves  
[jefferson.chaves@ifc-araquari.edu.br](mailto:jefferson.chaves@ifc-araquari.edu.br)

- Conhecer as formas de manipulação de um banco de dados;
- Executar as tarefas comuns em banco de dados:
  - CRIAR UM REGISTRO;
  - LER UM REGISTRO;
  - ATUALIZAR UM REGISTRO;
  - APAGAR UM REGISTRO;

A faint, dark green silhouette of an elephant is visible on the left side of the slide, partially obscured by the text.

**PROGRAMAÇÃO**

**BANCO DE DADOS**

## IMPORTANTE!

Não abordaremos detalhes de banco de dados, da linguagem SQL ou qualquer outro item referente a banco de dados. O objetivo do conteúdo a seguir é exclusivamente a integração do PHP com um SGBD.

- O PHP possui suporte a **12 sistemas gerenciadores de banco de dados (SGBD's)**;
- Podemos citar alguns SGBDs suportados pelo PHP:
  - PostgreSQL;
  - SQLite;
  - Oracle;
  - MySQL;

- O PHP permite a conexão e manipulação de bancos de dados;
- MySQL é o mais popular sistema de banco de dados usado com PHP;
- É um banco de dados relacional;
- **As interações com o banco são feitas por meio de instruções chamadas de consultas.**
- **O PHP oferece meios de executarmos tais consultas;**

1

2 //Exemplo de consulta:

3

4 **SELECT** nome, idade **FROM** nome\_tabela

5


6

7

8

9

10

A faint, dark green silhouette of an elephant is visible in the background on the left side of the slide.

**PROGRAMAÇÃO DE  
COMPUTADORES**

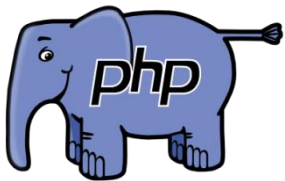
**CONEXÃO COM O  
SGBD**



- PHP e MySQL são tecnologias diferentes sendo executadas em ambiente diferentes;
- Para que seja possível manipular os dados de um banco de dados devemos primeiro **estabelecer uma conexão** entre essas tecnologias;

- O PHP permite algumas formas de conexão com o SGBD;
- A prática aconselhada é o uso da classe PHP Data Objects (**PDO**);

```
$conexao = new PDO("mysql:host=localhost;dbname=banco", usuario, senha)
```



`$conexao`



SGBD: MySQL

```
$conexao->exec( );  
$conexao->query( );
```



- A conexão com um banco de dados exige alguns parâmetros, conhecidos como DSN (data source name ):
  - **host**: o nome ou IP do servidor de bancos de dados;
  - **user**: o usuário no banco;
  - **password**: a senha para a conexão.

1

2

```
$conexao = new PDO("mysql:host=?;dbname=?", user, pass);
```

3

4

5

6

7

8

9

10

- Os blocos **Try** (tentar) e **Catch** (tratado) tem papel fundamental na conexão com o banco;
- **Ao tentar (try)** estabelecer uma conexão com o banco, podem ocorrer alguns erros gerados pelo SGBD como, banco inexistente, senha incorreta e etc;
- Nesses casos, uma exceção/erro é lançada e **capturada (catch)** facilitando a identificação do erro e seu tratamento.

```
1
2 try {
3     $conexao = new PDO("mysql:host;dbname", user, pass);
4 }
5 catch( PDOException $e ) {
6     echo "Conexão falhou: " . $e->getMessage( );
7 }
8
9
10
```

- Os blocos try e catch tratam apenas de erros na conexão;
- Para que todos os erros sejam tratados (erros em consultas, por exemplo) podemos opcionalmente configurar um atributo de modo de erro:



1

2 \$conexao->setAttribute(PDO::ATTR\_ERRMODE,  
3 PDO::ERRMODE\_EXCEPTION);

4

5

6

7


8

9

10

- Usando o a referencia \$conexão
- executar comando SQL por meio da variável de conexão;
- As duas principais funções para tanto são:
  - \$conexao->exec( );
  - \$conexao->query( );

- A função **\$conexao->exec( )** deve ser usada quando a consulta realizada não retorna registros algum, como um cadastro por exemplo;
- A função **\$conexao->query( )** deve ser usada quando solicitamos algum dados como por exemplo, uma lista dos produtos cadastrados;

A faint, dark green silhouette of an elephant is visible in the background on the left side of the slide.

**PROGRAMAÇÃO DE  
COMPUTADORES**

**ESTRUTURA DO  
BANCO**

- Para fins didáticos vamos adotar uma empresa hipotética chamada **LOJA IFC**;
- Se pensarmos em termos de estrutura, poderíamos organizar os dados de nossa empresa em:
  - **Funcionários;**
  - **Produtos;**
  - **Clientes;**
  - **Pedidos;**

- Devemos primeiro criar nosso banco de dados:
  - `$conexao->exec("CREATE DATABASE loja_ifc;");`
- Vamos definir na estrutura da tabela funcionários:
  - **Identificação;**
  - **Nome;**
  - **Sobrenome;**
  - **Email;**
  - **Data de cadastro;**


- E após definir os dados para representarmos um funcionário, podemos construir a estrutura da tabela funcionários;
- O comando SQL para criarmos uma tabela é o **CREATE TABLE nome\_tabela;**

```
1 CREATE TABLE funcionarios (  
2     id INT(6) AUTO_INCREMENT PRIMARY KEY,  
3     nome VARCHAR(30) NOT NULL,  
4     sobrenome VARCHAR(30) NOT NULL,  
5     email VARCHAR(50),  
6     data_cadastro TIMESTAMP  
7  
8  
9 )  
10
```



- **NOT NULL** – o campo não pode ficar vazio ou aceitar valores nulos;
- **DEFAULT** - valor padrão caso nenhum dados seja informado;
- **UNSIGNED** - restringe o uso apenas para números positivos e o 0;
- **AUTO INCREMENT** – soma 1 ao valor de um campo;
- **PRIMARY KEY** – usado para garantir um valor único a cada campo;

```
12 // IMPORTANTE: use exec() quando não houverem
13 //resultados retornados
14
15 // 1) Criar banco de dados
16 $sql = "CREATE DATABASE IF NOT EXISTS bd_loja_ifc;";
17 $conexao->exec($sql);
18
19 // *** Indicar qual banco usaremos
20 $sql = "USE bd_loja_ifc;";
21 $conexao->exec($sql);
22
23 // 2) Criar a tabela funcionarios
24 $sql = "CREATE TABLE IF NOT EXISTS tb_produtos (
25     id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
26     nome_produto VARCHAR(30) NOT NULL,
27     preco decimal(10,2) DEFAULT NULL,
28     descricao text,
29     categoria text
30 );";
31 $conexao->exec($sql);
```

A faint, dark green silhouette of an elephant is visible in the background on the left side of the slide.

**PROGRAMAÇÃO DE  
COMPUTADORES**

**INSERINDO  
REGISTROS**

- Depois de criarmos nosso **banco de dados** e nossa **tabela** podemos inserir e manipular dados cadastrados;
- Para inserirmos registros, algumas regras devem ser seguidas:
  - A consulta deve estar entre aspas duplas;
  - Variáveis ou valores devem estar entre aspas simples;

- A declaração **INSERT INTO** é usada para inserirmos registros em nosso banco de dados;
- **INSERT**  
**INTO nome\_tabela (coluna1, coluna2,...)**  
**VALUES (valor1, valor2,...)**
- Essas declarações são chamadas popularmente de **sql**.

- Para inserirmos novo registro na tabela, temos então que **estar conectados ao banco;**
- Precisamos de uma **variável de conexão** instanciada para acesso a outras funções deste banco;

## IMPORTANTE!

Quando uma coluna é marcada como **AUTO INCREMENT** ou como **TIMESTAMP** não é necessário declará-los em nossa sql. SGBD's fazem isso automaticamente;

# PROGRAMAÇÃO DE COMPUTADORES

# ACESSO AO BANCO DE DADOS

```
3  try
4  {
5      $conn = new PDO("mysql:host=$servidor;dbname=$dbname,
6          $username, $password");
7
8      // sql para criar a tabela funcionarios
9      $sql = "INSERT INTO funcionarios (nome, sobrenome, email)
10         VALUES ('José', 'Silva', 'jose.silva@email.com')";
11
12     // use exec() quando não houverem resultados retornados
13     $conn->exec($sql);
14
15     echo "Registro inserido com sucesso.";
16
17 } catch(PDOException $e) {
18
19     echo $sql . "<br>" . $e->getMessage();
20 }
```



- Quando executamos um comando **INSERT** ou **UPDATE** em uma tabela com **AUTO INCREMENT**, podemos acessar esse valor imediatamente com o método **lastInsertId( )**;

# PROGRAMAÇÃO DE COMPUTADORES

# ACESSO AO BANCO DE DADOS

```
3      try
4      {
5          $conn = new PDO("mysql:host=$servidor;dbname=$dbname,
6                          $username, $password");
7
8          // sql para criar a tabela funcionarios
9          $sql = "INSERT INTO funcionarios (nome, sobrenome, email)
10                VALUES ('José', 'Silva', 'jose.silva@email.com')";
11
12          // use exec() quando não houverem resultados retornados
13          $conn->exec($sql);
14
15          $ultimo_id = $conn->lastInsertId();
16          echo "Registro inserido com sucesso.";
17          echo "O último ID inserido foi: $ultimo_id";
18
19      } catch(PDOException $e) {
20          echo $sql . "<br>" . $e->getMessage();
21      }
```

A faint, dark green silhouette of an elephant is visible in the background on the left side of the slide.

**PROGRAMAÇÃO DE  
COMPUTADORES**

**LEITURA DE  
REGISTROS**

- Consulta é uma tarefa frequente na manipulação de banco de dados;
- Existem **N** formas de consultar um ou mais registros;
- Cada consulta variará a depender de sua complexidade. Exemplo:
  - Consultar o registro de um produto;
  - Consultar o registro de um amigo de um amigo meu.

- A declaração **SELECT FROM** é usada para selecionarmos registros em nosso banco de dados:
  - **SELECT campos FROM nome\_tabela;**
- Para executarmos essa consulta devemos usar a função query:
  - **\$consulta = \$conexao->query( );**
- Caso queiramos todos os campos, podemos substituir os campos por um asterisco;

- Ainda é necessário buscar esses dados usando a função **fetch** da conexão;
- **Fetch**: retorna apenas o primeiro registro;
- **FetchAll** : retorna todos os registros cadastrados;

```
3 try {  
4  
5     $conexao = new PDO("mysql:host=localhost;dbname=bd_loja",'root', 'root');  
6  
7     $consulta = $conexao->query("SELECT * FROM funcionarios");  
8  
9     $produtos = $consulta->fetchAll(PDO::FETCH_ASSOC);
```

- Leitura de um conjunto de registros:
- Para leitura de conjuntos de registros usamos a declaração **SELECT ... FROM ... WHERE**



```
try {  
  
    $conexao = new PDO("mysql:host=localhost;dbname=bd_loja_ifc",'root', 'root');  
  
    // Retorna uma declaracao: statement  
    $consulta = $conexao->query("SELECT * FROM tb_produtos WHERE id=$id");  
  
    $produto = $consulta->fetch(PDO::FETCH_ASSOC);  
  
}
```

1. Crie um formulário que contenha os campos carro, ano, preço.
2. Crie um banco de dados chamado bd\_automoveis
3. Insira os dados desse formulário no banco de dados por meio do comando INSERT INTO. Use como exemplo a inserção usadas nos arquivos instalar.php e conexao.php

A faint, dark green silhouette of an elephant is visible in the background on the left side of the slide.

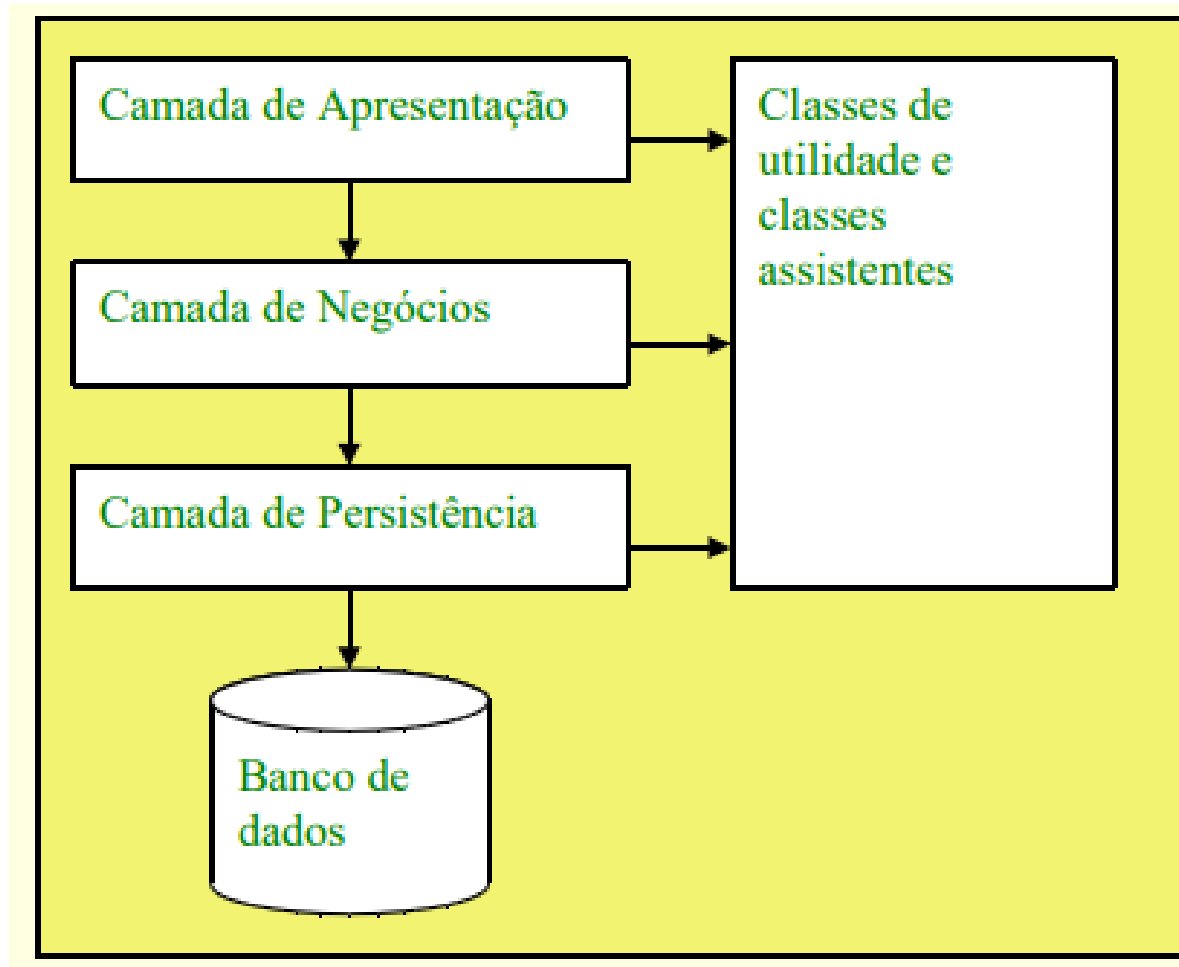
**PROGRAMAÇÃO DE  
COMPUTADORES**

**ARQUITETURA E  
PADRÕES E OUTRAS  
COISAS**

- Em aplicações POO diversos aspectos devem ser considerados:
  - Apresentação Lógica da aplicação;
  - Lógica do negócio;
  - Persistência de dados (capacidade de uma aplicação manter suas informações entre sessões de uso);
  - Camada de Utilitários: Controle de Exceções, Logging, comunicação, etc.

- Uma proporção significativa do esforço de desenvolvimento recai sobre como organizar uma solução aproximadamente ótima para um problema.

- Arquitetura em camadas:
  - visa a **criação de aplicações modulares**, de forma que a camada mais alta se comunique com a camada mais baixa e assim por diante, fazendo com que uma camada seja dependente apenas da camada imediatamente abaixo.



- **Camada de Apresentação:** Lógica de interface do usuário (GUI). O código responsável pela apresentação e controle da página e tela de navegação forma a camada de apresentação;
- **Camada de Modelo:** Código referente a implementação de regras de negócio ou requisitos do sistema;
- **Camada de persistência:** Responsável por armazenamento e recuperação dos dados quando solicitado. Objetivo é o de garantir uma independência da fonte de dados (arquivos, bancos de dados).



- **Banco de dados:** O BD existe fora da aplicação. Essa camada deve representar a conexão entre o banco e a aplicação;
- **Assistentes e Classes de utilidade:** São classes necessária para o funcionamento ou mesmo o complemento de uma aplicação ou parte dela, como por exemplo o ***Exception*** para tratamento de erros.

A faint, dark silhouette of an elephant is visible in the background on the left side of the slide.

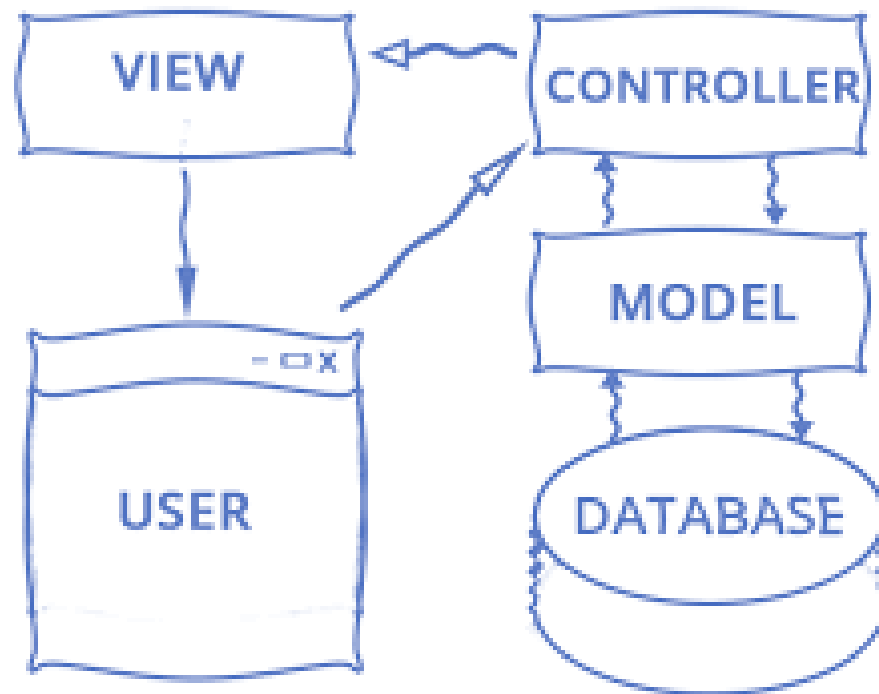
**PROGRAMAÇÃO DE  
COMPUTADORES**

**MODEL VIEW  
CONTROLLER - MVC**

- Usado no desenvolvimento de aplicações para facilitar o desenvolvimento em camadas de aplicações que usam a orientação a objetos.
- Seu objetivo é isolar as camadas, evitando que a mudança em uma camada afete outra;
- Facilita o desenvolvimento por times multidisciplinares.

# PROGRAMAÇÃO DE COMPUTADORES

# ARQUITETURA EM CAMADAS



- **Model:** Aqui devem estar as regras de negócio. Validar um CPF, vender um produto, verificar usuário logado são algoritmos que devem estar nessa camada.
- **View:** Responsável por gerar a forma como a resposta será apresentada, página *web*, formulário, relatório, etc.
- **Controller:** Responsável por responder aos pedidos por parte do utilizador. Sempre que um utilizador faz um pedido ao servidor esta camada é a primeira a ser executada.