**APP DASHBOARD – AJAX**

**Ajax com jQuery – Conceitos básicos / iniciando o App Dashboard**

**Ajax** – Asynchrounous JS and XML.

**XMLHttpRequest** – objeto que pode ser instanciado de forma nativa através da linguagem JS.

Dshboard

Aplicação front-end;

Irá consumir dados do back-end através de requisições assíncronas;

PHP;

Banco de dados MySQL.

**Requisições para o back-end com load(), get() e post()**

Funcionam como uma interface para a manipulação do objeto XMLHttpRequest;

Objeto utilizado para realizar e controlar requisições HTTP, é como uma espécie de facilitador desse processo;

Através desses métodos podemos fazer essas requisições de forma enxuta, prática.

<li><a href="#" id="documentacao">

<li><a href="#" id="suporte">

Adicionando id aos links

$('#documentacao').on('click', ()=>{

        console.log('link documentacao clicado')

    })

    $('#suporte').on('click', ()=>{

        console.log('link suporte clicado')

    })

Testando links

**Método load()**

$("#documentacao").on("click", () => {

    $('#pagina').load('documentacao.html')

  });

  $("#suporte").on("click", () => {

    $('#pagina').load('suporte.html')

  });

Assim que elemento com id documentação for clicado, página recebe conteúdo do arquivo ‘documentação.html’;

Ocorre uma sobreposição do conteúdo;

Através de poucas linhas de código, foi possível fazer um Request.

    $.get("documentacao.html", (data) => {//podereia ser qualquer parâmetro (está sendo recuperado o conteúdo em si da resposta) - data, valor..

através da instancia do jQuery ($) será executado o método get (.get()) que espera a url .get('') e uma ação .get('', data => {}) - data pois estamos recuperando o conteúdo em si da resposta.

  $.post("suporte.html", (data) => {

      $("#pagina").html(data);

    });

Post é feito de forma análoga ao get

**Preparando o back-end da aplicação**

Criação do arquivo **app.php** que será responsável por se **conectar com o banco de dados**;

**Através do front-end** da aplicação utilizando Ajax com jQuery **serão feitas requisições assíncronas para** esse script (**app.php**) com objetivo de **obter os registros do banco de dados**, **recuperar** essa **informação** e **atualizar o front-end** dessa aplicação **sem que haja** um **refresh** na página;

Processo de recuperação de dados e exibição dos mesmos será feito internamente na aplicação front-end.

Dentro do app.php, será criada uma classe dashboard, com base nessa classe, será instanciado um objeto que contenha todos os atributos necessários para popular os dados do nosso dashboard.

Em **app.php**

//classe dashboard

    //atributos importantes que definem a competência do dashboard

//classe de conexao com o banco

private function conectar() //método público de conexão

    {

        try {

            $conexao = new PDO(

                //adicionando a variável $conexao a instancia de PDO

                "mysql:host=$this->host;dbname=$this->dbname",

                //prâmetros de conexão do BD - host + nome do bd

                "$this->user",

                //user

                "$this->pass" //senha

            );

            //como está sendo trabalhado com o BD e o front-end que utilizam o UTF-8,

            //setando que a comunicação entre o back-end e BD também deve ser tratado através do utf-8

            $conexao->exec('set charset utf8');

            //variável que contém a instância da conexão + exec(do próprio PDO) que executa a instrução para que a instância da conexão trabalhe comm utf-8

            return $conexao;

        } catch (PDOException $e) { //caso ocorra algum erro, recupera a mensagem do mesmo

            echo '<p>' . $e->getMessage() . '</p>';

        }

    }

//classe (model) --- irá permitir a manipulação do objeto no banco, como se fosse o model para trabalharmos com o projeto

    public function \_\_construct(Conexao $conexao, Dashboard $dashboard) //método construct recebe conexao e dashboard e são tipados, pois são criados baseados em classes já criadas

        $this->conexao = $conexao->conectar(); //retornando conexão e método conectar

    public function getNumeroVendas()//será responsável por recuperar o indicador do número de vendas no BD

  {

        $query = '';//é necessário criar uma query, e ela deve ser uma instrução SQL válido para o server de BD

        $stmt = $this->conexao->prepare($query);//preparo da query - retorna o stmt

        $stmt->bindValue();

        $stmt->execute();//execução da query

        return $stmt->fetch(PDO::FETCH\_OBJ);//retorna valor que foi recuperado do banco de dados como objeto

    }

  $query = 'select count(\*) as numero\_vendas from tb\_vendas where data\_venda between :data\_inicio AND :data\_fim';

  $stmt->bindValue(':data\_inicio', '2018-10-01');

  $stmt->bindValue(':data\_fim', '2018-08-23');

$dashboard->\_\_set('numeroVendas', $db->getNumeroVendas()); //setando atributo numeroVendas com base no retorno que obtemos de getNumeroVendas

        return $stmt->fetch(PDO::FETCH\_OBJ)->numero\_vendas;

Setando apenas o numero\_vendas exibe apenas essa informação, ao invés de retornar o objeto por completo.

$stmt->bindValue(':data\_inicio', $this->dashboard->\_\_get('data\_inicio'));

        //recuperando através do this o arquivo dashboard que contém o objeto e através do método get(interno do dashboard), recuperamos o atributo data\_inicio e data\_fim

        $stmt->bindValue(':data\_fim', $this->dashboard->\_\_get('data\_fim'));

Informações já estão dinâmicas

public function getTotalVendas() //será responsável por recuperar o indicador do número de vendas no BD

    {

        $query = 'select SUM(total) as total\_vendas from tb\_vendas where data\_venda between :data\_inicio AND :data\_fim'; //é necessário criar uma query, e ela deve ser uma instrução SQL válido para o server de BD

        $stmt = $this->conexao->prepare($query); //preparo da query - retorna o stmt

        $stmt->bindValue(':data\_inicio', $this->dashboard->\_\_get('data\_inicio'));

        //recuperando através do this o arquivo dashboard que contém o objeto e através do método get(interno do dashboard), recuperamos o atributo data\_inicio e data\_fim

        $stmt->bindValue(':data\_fim', $this->dashboard->\_\_get('data\_fim'));

        $stmt->execute(); //execução da query

        return $stmt->fetch(PDO::FETCH\_OBJ)->total\_vendas; //retorna valor que foi recuperado do banco de dados como objeto

    }

Calculando valor total das vendas

print($bd->getTotalVendas());