Memoria Sección 6_7_8
COW

Nombre: Muhammad Mohsin Zaman Shaheen
Fecha: 11/5/2024

Apartado 1 (JQuery & JQuery Widgets)

En este apartado el objetivo es familiarizarnos con JQuery y JQwidgets. Para cumplir con eso,he realizado varios cambios y nuevas funcionalidades. Para empezar, el primer cambio es el siguiente donde he cambiado la funcionalidad de autocompletar realizada en la práctica anterior con Ajax que ahora sigue funcionando de la misma forma pero la implementación es con el método ".ajax" de JQuery en vez de utilizar "XMLHttpRequest()" .

A continuación se puede observar el cambio:

```
function selectSuggestion(suggestion) {
    document.getElementById('destinationInput').value = suggestion;
    document.getElementById("hintContainer").innerHTML = "";
}

function showHint(str) {
    if (str.length == 0) {
        document.getElementById("hintContainer").innerHTML = "";
        return;
    } else {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
            if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
                var suggestions = xmlhttp.responseText;
                var hintContainer = document.getElementById("hintContainer");
                if (suggestions === "no suggestion") {
                    hintContainer.innerHTML = "";
                } else {
                    // Populate suggestion box with suggestions
                    hintContainer.innerHTML = "<div class='suggestion-box'>" + suggestions + "</div>";
                    var suggestionItems = document.querySelectorAll(".suggestion-box .suggestion-item");
                    suggestionItems.forEach(function(item) {
                        item.addEventListener("click", function() {
                            selectSuggestion(item.innerText);
                        });
                    });
                }
            }
        };
        xmlhttp.open("GET", "./php/searchHint.php?q=" + str, true);
        xmlhttp.send();
    }
}
```

Autocomplete con AJAX(XMLHttpRequest)

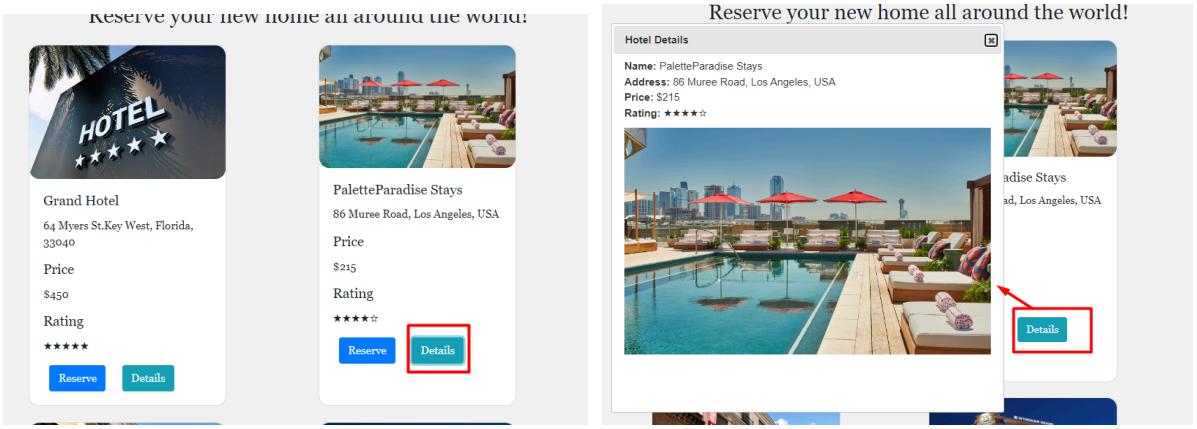
```
$(document).ready(function() {
    $('#destinationInput').on('keyup', function() {
        var input = $(this).val();
        if (input.length == 0) {
            $('#hintContainer').empty();
        } else {
            $.ajax({
                url: './php/searchHint.php',
                type: 'GET',
                data: {
                    q: input
                },
                success: function(data) {
                    if (data === 'no suggestion') {
                        $('#hintContainer').empty();
                    } else {
                        $('#hintContainer').html("<div class='suggestion-box'>" + data + "</div>");
                        $('.suggestion-box .suggestion-item').on('click', function() {
                            $('#destinationInput').val($(this).text()).focus();
                            $('#hintContainer').empty();
                        });
                    }
                }
            });
        }
    });
});
```

Autocomplete con JQuery

JQuery Widgets

Dialog Box

Para aprender JQuery widgets, he decidido integrar el siguiente widget que trata de un dialog box en mi webapp. El objetivo de este dialog box es mostrar detalles adicionales de un hotel. Para abrir el dialog box he creado un botón de “details” Los datos que se muestran se obtienen desde un JSON usando el método *“getJSON”* de Javascript. A continuación podemos observar el resultado final:



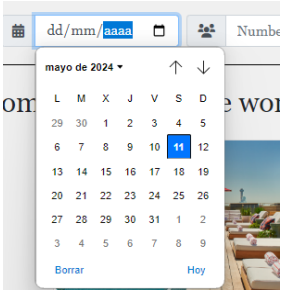
A continuación se puede ver el código correspondiente a esta nueva funcionalidad para mostrar el diálogo Box. También he añadido algunos *efectos visuales* como del fade-in o fade-out al Dialog-Box.

```
$(document).ready(function() {
    $("#detailsDialog").dialog({
        autoOpen: false,
        width: 600, // Width of the dialog
        height: 600, // Height of the dialog
        modal: true,
        resizable: true, // Prevent resizing
        draggable: true, // Allow moving the dialog
        title: 'Hotel Details', // Title of the dialog
        show: {
            effect: "fade",
            duration: 300
        },
        hide: {
            effect: "fade",
            duration: 300
        }
    });
```

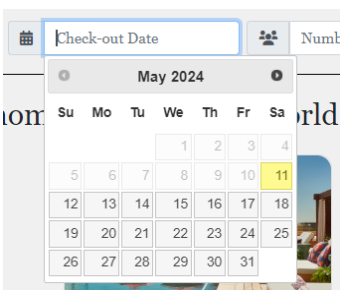
```
$(".details-button").click(function() {
    var index = $(this).data('index'); // Get the index of the hotel from the data-index attribute
    $.getJSON('./data/hotels.json', function(data) {
        var hotel = data[index];
        var detailsHtml = '<div><strong>Name:</strong> ' + hotel.name +
            '</div><div><strong>Address:</strong> ' + hotel.address +
            '</div><div><strong>Price:</strong> ' + hotel.price +
            '</div><div><strong>Rating:</strong> ' + hotel.rating +
            '</div><img src="' + hotel.image + '" style="width:100%; margin-top:10px;">';
        $("#detailsDialog").html(detailsHtml).dialog("open");
    });
});
```

Date Picker

Otro de los widgets interesantes que he implementado es el Datepicker de JQuery sustituyendo por el default de bootstrap que tenía antes. A continuación observamos la diferencia:



Antes



Despues

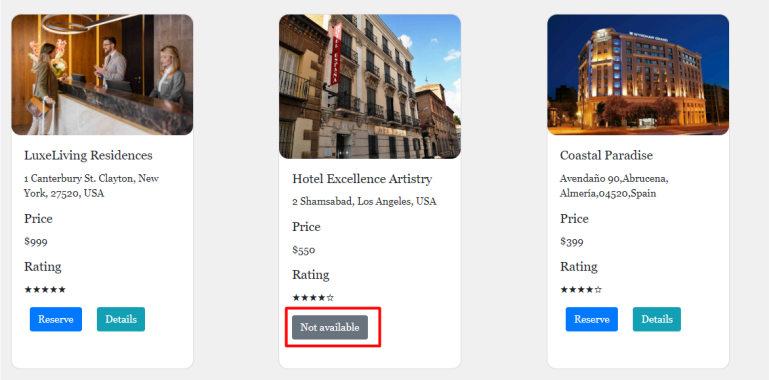
Para realizar el cambio he implementado el siguiente código con la función *“datepicker”:*

```
$(function() {
    $('#checkinDateInput, #checkoutDateInput').datepicker({
        // Set the minimum date to today
        minDate: new Date(),
        dateFormat: 'yy-mm-dd'
    });
});
```

Apartado 2 (Datos con JSON)

En esta sección el objetivo es interactuar con JSON de manera que primeramente hay que recuperar datos desde la base de datos, guardarlos en un JSON. Una vez tenemos eso, pasamos esa información al cliente que extraerá(fetch) la información desde el JSON y lo mostrará en una tabla que se crea dinámicamente.

En mi caso, como podemos ver, un usuario puede ver los hoteles que estas disponibles para reservar o no tal que así:



Lo que pretendo implementar es que el usuario actual pueda ver las fecha en la que ese hotel no está disponible y por quien está reservado:

A continuación muestro todos estos pasos y el resultado final:

1) ***Cliente:*** Un listener que entra en acción al pasar el ratón por el botón de “Not Available” y hace el “fetch” de los datos desde json.

```
document.addEventListener('DOMContentLoaded', function() {
    const unavailableButtons = document.querySelectorAll('.not-available');

    unavailableButtons.forEach(button => {
        let detailsDiv = null;

        button.addEventListener('mouseenter', function() {
            if (!detailsDiv) {
                detailsDiv = document.createElement('div');
                detailsDiv.className = 'reservation-details';
                this.appendChild(detailsDiv);
                fetchDetails(this.getAttribute('data-hotel'), detailsDiv);
            }
            detailsDiv.style.display = 'block';
        });

        button.addEventListener('mouseleave', function() {
            if (detailsDiv) {
                detailsDiv.style.display = 'none';
            }
        });
    });
});
```

```
function fetchDetails(hotelName, detailsDiv) {
    fetch(`./php/getReservationDetails.php?hotelName=${encodeURIComponent(hotelName)}`)
        .then(response => response.json())
        .then(data => {
            if (data) {
                displayDetails(data, detailsDiv);
            } else {
                console.log("No data available or data is empty:", data);
            }
        })
        .catch(error => console.error('Error fetching data:', error));
}
```

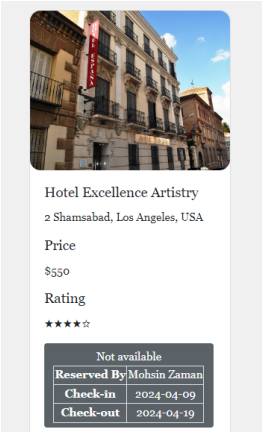
2) ***Servidor:*** El “fetch” llama al servidor que es el que recupera los datos de la Base de datos y guarda en formato JSON.

```php
<?php
session_start();
require('./db_management/db_connection.php');

$hotelName = urldecode($_GET['hotelName']);

try {
    $stmt = $conn->prepare("SELECT name, checkin_date, checkout_date FROM reservations WHERE hotel_name = :hotelName");
    $stmt->bindParam(':hotelName', $hotelName, PDO::PARAM_STR);
    $stmt->execute();
    $reservationDetails = $stmt->fetch(PDO::FETCH_ASSOC);
    header('Content-Type: application/json');
    echo json_encode($reservationDetails);
} catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
```

getReservationDetails.php

Mostrar desde Json en tabla creada dinamicamente

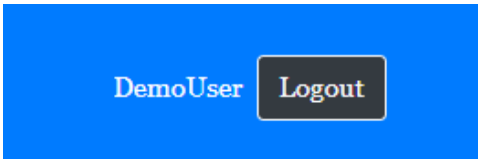
```javascript
function displayDetails(data, detailsDiv) {
    let tableHTML = `<table class="table-bordered">
                <tr>
                    <th>Reserved By</th>
                    <td>${data.name}</td>
                </tr>
                <tr>
                    <th>Check-in</th>
                    <td>${data.checkin_date}</td>
                </tr>
                <tr>
                    <th>Check-out</th>
                    <td>${data.checkout_date}</td>
                </tr>
            </table>`;
    detailsDiv.innerHTML = tableHTML;
}
```



Apartado 3 (Datos con XML)

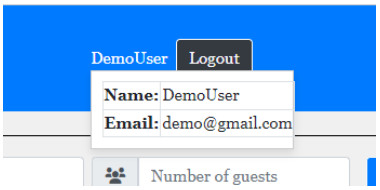
En este apartado el objetivo es muy similar al apartado anterior pero esta vez en vez de JSON, hay que utilizar XML y mostrar los datos en una tabla creada dinámicamente. Cabe destacar que los datos deben ser diferentes a los del apartado anterior por lo que esta vez voy a mostrar la información del usuario que está logeado.

Para eso, he decidido añadir una funcionalidad nueva que trata de mostrar el nombre del usuario que está logueado:



Ahora, para cumplir este objetivo, he decidido mostrar información de este usuario al hacer hover sobre el nombre de usuario:

On hover:



Para conseguir eso de forma similar a antes, he recuperado los datos del usuario pero esta vez en vez de enviar al cliente como Json, envio como XML como se ve a continuación:

```php
$result = $stmt->fetch(PDO::FETCH_ASSOC);
if ($result) {
    $xml = new SimpleXMLElement('<UserDetails/>');
    foreach ($result as $key => $value) {
        $xml->addChild($key, htmlspecialchars($value));
    }
    Header('Content-type: text/xml');
    echo $xml->asXML();
```

En el Cliente, utilizó la función “fetch” de Javascript para extraer los datos y mostrarlo en una tabla creada dinámicamente:

```javascript
userNameDisplay.addEventListener('mouseenter', function() {
    fetch('./php/getUserDetails.php')
        .then(response => response.text())
        .then(str => (new window.DOMParser()).parseFromString(str, "text/xml"))
        .then(data => {
            const name = data.getElementsByTagName('name')[0].textContent;
            const email = data.getElementsByTagName('email')[0].textContent;
            userDetails.innerHTML = `<table class="table-bordered">
                                        <tr><th>Name:</th><td>${name}</td></tr>
                                        <tr><th>Email:</th><td>${email}</td></tr>
                                    </table>`;
            const rect = userNameDisplay.getBoundingClientRect();
            userDetails.style.top = `${rect.bottom + window.scrollY}px`;
            userDetails.style.left = `${rect.left + window.scrollX}px`;
            userDetails.style.display = 'block';
        })
        .catch(error => console.error('Error fetching user details:', error));
});
```