

EVALUACIÓN

PROGRAMACIÓN WEB II Semana 3

David Vera
24/09/24
Ingeniería en Informática



1. Escribe el fragmento de código JavaScript y los ajustes realizados al HTML, con el fin de crear una interfaz de usuario interactiva que permita al usuario filtrar destinos y fechas de viaje, y actualizar dinámicamente los resultados en la página.

Para lograr una interfaz de usuario interactiva se ha modificado el código de ejemplo, añadiendo los campos y botones necesarios. Asimismo, una separación entre la búsqueda de vuelos con la de paquetes mediante *tabs* de bootstrap. Sólo incluiré las líneas de código más relevantes en este informe, sin embargo puede revisar el proyecto completo en el repositorio creado para su fin en:

<https://github.com/dsveraa/js-flight-schedule>

Respecto al filtrado, la página sugiere orígenes y destinos de acuerdo a un *datalist* preparado para dicha tarea:

Búsqueda de Vuelos

Búsqueda de Paquetes

Búsqueda de vuelos

Seleccione su origen, destino y fecha del vuelo:

Buscar Vuelos

Limpiar

Brasilia

Lima

Madrid

Miami

```
<input
  type="text"
  id="origin-f"
  list="cities-origin"
  placeholder="Origen"
  size="9"
/>
<datalist id="cities-origin">
  <option value="Buenos Aires">Buenos Aires</option>
  <option value="Brasilia">Brasilia</option>
  <option value="Lima">Lima</option>
  <option value="Madrid">Madrid</option>
  <option value="Miami">Miami</option>
  <option value="Santiago">Santiago</option>
</datalist>
```

Esto se complementa con funciones que permiten la omisión de sugerencias de ciudades, si alguna ya ha sido seleccionada, mediante la reescritura del elemento *option* de manera dinámica:

```
function filterDestinations(inputId, datalistId) {
  const origin = document.getElementById(inputId).value
  const datalist = document.getElementById(datalistId)

  datalist.innerHTML = ""

  cities.forEach((city) => {
    if (city !== origin) {
      const option = document.createElement("option")
      option.value = city
      datalist.appendChild(option)
    }
  })
}
```

Con su aplicación:

```
document.getElementById("origin-f").addEventListener("input", function () {  
  filterDestinations("origin-f", "cities-destination")  
})
```

Una validación básica:

```
if (!origin || !destination || !travelDate) {  
  resultsContainer.classList = "alert alert-warning"  
  resultsContainer.innerHTML =  
    "Falta información, todos los campos son obligatorios."  
  return  
}
```

Seleccione su origen, destino y fecha del vuelo:



Buscar Vuelos

Limpiar

Falta información, todos los campos son obligatorios.

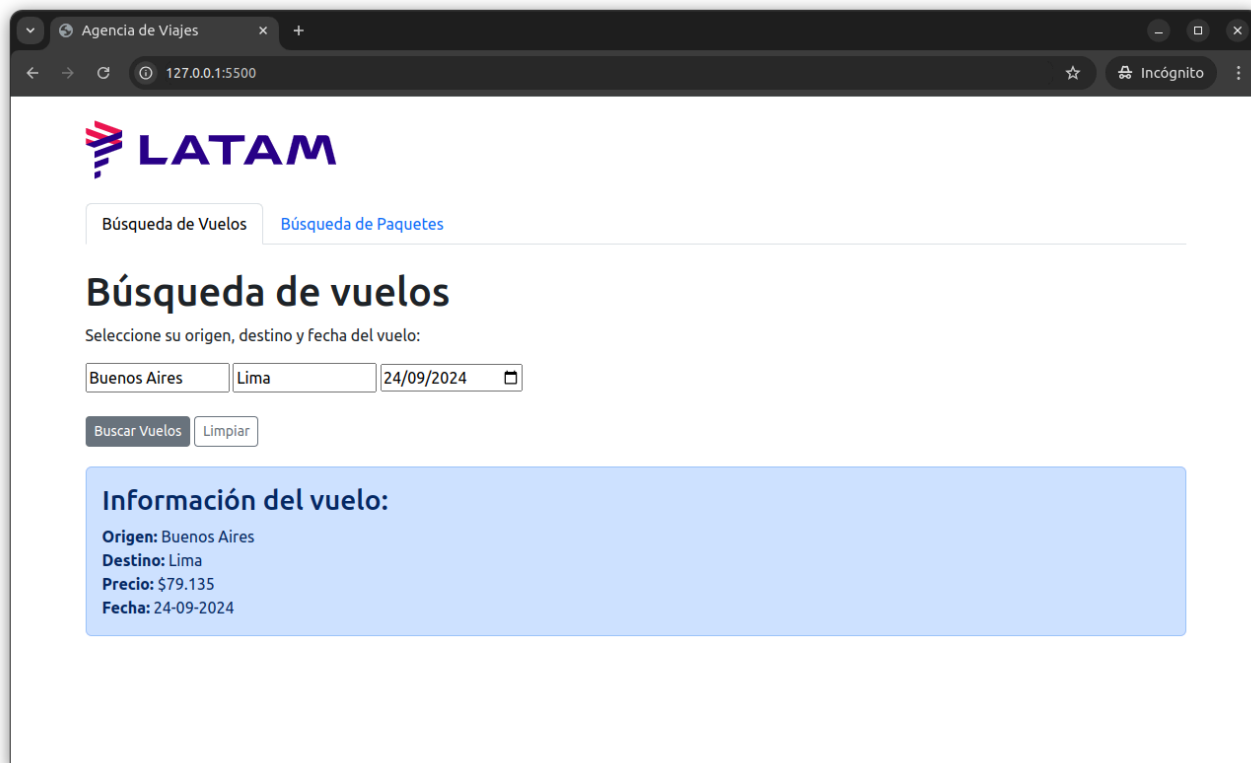
Los datos son comparados con un archivo JSON que cumple en el rol de nutrir de información a la app, y son filtrados para obtener el resultado esperado en caso de que exista:

```
loadFlightsFromJSON().then((flights) => {  
  const filteredFlights = flights.filter(  
    (flight) =>  
      flight.origin.toLowerCase().includes(origin.toLowerCase()) &&  
      flight.destination.toLowerCase().includes(destination.toLowerCase()) &&  
      flight.dates.includes(travelDate)  
  )  
})
```

Si hay resultados, se pasa la información al DOM mediante un loop, en caso de existir más de uno:

```
if (filteredFlights.length > 0) {  
  filteredFlights.forEach((flight) => {  
    const resultDiv = document.createElement("div")  
    resultDiv.innerHTML = `  
      <h3>Información del vuelo:</h3>  
      <strong>Origen:</strong> ${flight.origin}<br>  
      <strong>Destino:</strong> ${flight.destination}<br>  
      <strong>Precio:</strong> ${flight.price.toLocaleString(  
        "es-ES"  
      )}<br>  
      <strong>Fecha:</strong> ${esDate}  
    `;  
    resultsContainer.appendChild(resultDiv)  
    resultsContainer.classList = "alert alert-primary"  
  })  
}
```

Obteniendo por ejemplo lo siguiente:




The screenshot shows a web browser window with the URL 127.0.0.1:5500. The page is the LATAM flight search interface. It has a header with the LATAM logo and two tabs: "Búsqueda de Vuelos" (selected) and "Búsqueda de Paquetes". Below the tabs, the title "Búsqueda de vuelos" is displayed. A prompt "Seleccione su origen, destino y fecha del vuelo:" is followed by three input fields: "Buenos Aires", "Lima", and "24/09/2024" with a calendar icon. Below these fields are two buttons: "Buscar Vuelos" and "Limpiar". A light blue box titled "Información del vuelo:" contains the following details: "Origen: Buenos Aires", "Destino: Lima", "Precio: \$79.135", and "Fecha: 24-09-2024".

Y si el vuelo no existe, tiene el siguiente comportamiento:

```
} else {  
  resultsContainer.classList = "alert alert-danger"  
  resultsContainer.innerHTML = `No hay vuelos de <strong>${origin}</strong> a  
  <strong>${destination}</strong> en esta fecha.`  
}  
})  
}
```

Búsqueda de vuelos

Seleccione su origen, destino y fecha del vuelo:



No hay vuelos de **"Santiago"** a **"Miami"** en esta fecha.

2. Escribe el fragmento de código JavaScript correspondiente al sistema de objetos diseñado que represente diferentes paquetes turísticos, con sus propiedades y métodos que faciliten su gestión y visualización en la página web.

En esta parte, seguimos usando los mismos datos de vuelos que rescatamos del JSON, pero modelamos la información de hoteles y paquetes turísticos mediante clases:

```
class Hotel {  
  constructor(name, location, nightPrice) {  
    this.name = name  
    this.location = location  
    this.nightPrice = nightPrice  
  }  
}
```

```
class Package {  
  constructor(flight, hotel, nights) {  
    this.flight = flight  
    this.hotel = hotel  
    this.nights = nights  
  }  
}
```

La clase Package tiene un método para calcular el precio del paquete:

```
calculatePrice() {  
  const flightPrice = this.flight.price  
  const hotelPricePerNight = this.hotel.nightPrice  
  const hotelTotalPrice = hotelPricePerNight * this.nights  
  const totalPrice = flightPrice + hotelTotalPrice  
  return totalPrice  
}
```

Y otro para devolver la información procesada:

```
showPackageDetails() {  
  return `  
    <h3>Paquete turístico:</h3>  
    <strong>Origen</strong>: ${this.flight.origin}<br>  
    <strong>Destino:</strong> ${this.flight.destination}<br>  
    <strong>Precio del vuelo:</strong> ${this.flight.price.toLocaleString("es-ES")}<br>  
    <strong>Hotel:</strong> ${this.hotel.name}<br>  
    <strong>Ubicación del hotel:</strong> ${this.hotel.location}<br>  
    <strong>Precio por noche:</strong> ${this.hotel.nightPrice.toLocaleString("es-ES")}<br>  
    <strong>Número de noches:</strong> ${this.nights}<br>  
    <strong>Precio total del paquete:</strong> ${this.calculatePrice().toLocaleString("es-ES")}  
  `;  
}
```


Dentro de la función para buscar paquetes turísticos, se cargan los vuelos, luego se definen los hoteles como instancias de la clase Hotel por ciudad y se filtra la búsqueda de acuerdo a los datos disponibles:

```
function searchPackages() {  
  loadFlightsFromJSON().then((flights) => {  
    const hotels = {  
      Santiago: new Hotel("Ibis Providencia", "Santiago", 56170),  
      Lima: new Hotel("JW Marriott Hotel Lima", "Lima", 38140),  
      "Buenos Aires": new Hotel("Alvear Palace Hotel", "Buenos Aires", 26823),  
      Brasilia: new Hotel("Royal Tulip Brasilia Alvorada", "Brasilia", 41235),  
      Madrid: new Hotel("Hotel Ritz Madrid", "Madrid", 85125),  
      Miami: new Hotel("The Ritz-Carlton, South Beach", "Miami", 78255),  
    }  
  
    const filteredFlights = flights.filter(  
      (flight) =>  
        flight.origin.toLowerCase().includes(origin.toLowerCase()) &&  
        flight.destination.toLowerCase().includes(destination.toLowerCase()) &&  
        flight.dates.includes(travelDate)  
    )  
  })  
}
```

Si hay coincidencias, se itera la lista creando dinámicamente un paquete con la información del vuelo, hotel y noches, para luego traspasar todo al DOM procesado gracias al método de la clase:

```
if (filteredFlights.length > 0) {  
  filteredFlights.forEach((flight) => {  
    const hotel = hotels[flight.destination]  
    if (hotel) {  
      const pkg = new Package(flight, hotel, nights)  
      const resultDiv = document.createElement("div")  
      resultDiv.innerHTML = pkg.showPackageDetails()  
      resultsContainer.classList = "alert alert-primary"  
      resultsContainer.appendChild(resultDiv)  
    }  
  })  
}
```

Búsqueda de paquetes turísticos

Seleccione su origen, destino, fecha del vuelo y cantidad de noches de hotel:

<input type="text" value="Brasilia"/>	<input type="text" value="Madrid"/>	<input type="text" value="25/09/2024"/>	<input type="text" value="4"/>
---------------------------------------	-------------------------------------	---	--------------------------------

Paquete turístico:

Origen: Brasilia
Destino: Madrid
Precio del vuelo: \$250.999
Hotel: Hotel Ritz Madrid
Ubicación del hotel: Madrid
Precio por noche: \$85.125
Número de noches: 4
Precio total del paquete: \$591.499

3. Escribe el fragmento de código JavaScript donde se implementaron eventos que muestren notificaciones instantáneas sobre ofertas especiales y actualizaciones en tiempo real sobre la disponibilidad de paquetes turísticos.

Para mostrar información relacionada a paquetes turísticos, quise que no fuera algo molesto, como un `alert()`, que está prohibido si hablamos de buenas prácticas en la experiencia de usuario en este contexto.

Entonces cada vez que se haga una búsqueda, el programa buscará un destino alternativo para sugerir un nuevo paquete turístico de forma aleatoria:

```
function searchPackages() {  
  loadFlightsFromJSON().then((flights) => {  
    const alternativeFlights = flights.filter(  
      (flight) =>  
        flight.origin.toLowerCase().includes(origin.toLowerCase()) &&  
        flight.dates.includes(travelDate) &&  
        flight.destination.toLowerCase() !== destination.toLowerCase()  
    )  
  
    if (alternativeFlights.length > 0) {  
      const randomFlight = getRandomElement(alternativeFlights)  
      const altHotel = hotels[randomFlight.destination]  
      if (altHotel) {  
        const altPkg = new Package(randomFlight, altHotel, nights)  
  
        showSimpleNotification(  
          `¿Qué tal un viaje a ${  
            randomFlight.destination  
          } en las mismas fechas por ${altPkg  
            .calculatePrice()  
            .toLocaleString("es-ES")}?`,  
          randomFlight,  
          altHotel,  
          nights  
        )  
      }  
    }  
  })  
}
```

Paquete turístico:

Origen: Brasilia
Destino: Madrid
Precio del vuelo: \$250.999
Hotel: Hotel Ritz Madrid
Ubicación del hotel: Madrid
Precio por noche: \$85.125
Número de noches: 4
Precio total del paquete: \$591.499

¿Qué tal un viaje a Miami en las mismas fechas por \$763.785?

[Ver detalles](#)

<https://dsveraa.github.io/js-flight-schedule/> para una experiencia interactiva de la tarea semana 3.

REFERENCIAS BIBLIOGRÁFICAS

IACC (2022). *Introducción al DOM*. Programación Web II. Semana 3

IACC (2022). *Objetos y Eventos en JavaScript*. Programación Web II. Semana 3