**Amadeus paginas**

<https://developers.amadeus.com/blog/airport-autocomplete-app-with-the-mern-stack> (Example)

Estilos de módulos en JavaScript: **ESM (ECMAScript Modules)** y **CommonJS**.

**CommonJS (require/exports)**

const express = require("express");

require('dotenv').config({ path: '.env.local' });

module.exports = router; const router = require("./router");

module.exports = { CLIENT\_ID: process.env.API\_KEY,

CLIENT\_SECRET: process.env.API\_SECRET, };

**ESM (import/export)**:

En el package.json: "type": "module",

import express from "express";

import dotenv from "dotenv";

dotenv.config({ path: '.env.local' });

export default router; import router from "./router.js"

export { CLIENT\_ID, CLIENT\_SECRET }; import { CLIENT\_ID, CLIENT\_SECRET } from "./config.js";

export const CLIENT\_SECRET = process.env.API\_SECRET;

**Obtener accestoken de amadeus en git Bash**

curl -k "https://test.api.amadeus.com/v1/security/oauth2/token" \ -H "Content-Type: application/x-www-form-urlencoded" \ -d "grant\_type=client\_credentials&client\_id=Y6oLQH86wPd5Y9VVw38ACfT0hVqfY5Pp&client\_secret=aW7e5naDRVjdORFI"

Npx create-react-app airline-web

install react-router-dom

npm install react-icons –save

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(

    <BrowserRouter>

      <App />

    </BrowserRouter>

);

**BackEnd**

**nodemon**

"scripts": { "start": "node index.js",

"dev": "nodemon index.js"

**Instalar nodemon**

El script dev que configuramos usa nodemon, una herramienta que reinicia automáticamente tu servidor cada vez que detecta cambios en tus archivos de código. Esto es útil para el desarrollo, ya que no necesitas reiniciar el servidor manualmente después de cada cambio.

1. **Instalar nodemon**:

bash

Copy code

npm install nodemon --save-dev

* + **--save-dev** asegura que nodemon se instala como una dependencia de desarrollo y no como parte de las dependencias de producción.

const { page, subType, keyword } = req.query;

**req.query**: Esto extrae los parámetros de la consulta (query parameters) de la URL. Por ejemplo, si la URL es /api/airports?page=1&subType=AIRPORT&keyword=LAX, entonces:

* page será "1"
* subType será "AIRPORT"
* keyword será "LAX"

 const response = await amadeus.client.get("/v1/reference-data/locations", {

    keyword,

    subType,

    "page[offset]": page \* 10

  });

 **amadeus.client.get(...)**: Este es un método proporcionado por la biblioteca de Amadeus para realizar una solicitud GET a la API de Amadeus.

 **/v1/reference-data/locations**: Es el endpoint de Amadeus que se está llamando. Este endpoint se utiliza para buscar ubicaciones, como aeropuertos o ciudades.

 **Parámetros de la Solicitud**:

* keyword: Este es el parámetro de búsqueda, como un código IATA (por ejemplo, "LAX").
* subType: Es el tipo de ubicación que se está buscando, como "AIRPORT" o "CITY".
* "page[offset]": page \* 10: Esto configura el paginado de los resultados. page es multiplicado por 10 para calcular el desplazamiento en la paginación.

 const parsedBody = JSON.parse(response.body);

 **response.body es una Cadena de Texto**:

* Cuando haces una solicitud a la API de Amadeus utilizando amadeus.client.get(...), la propiedad response.body contiene la respuesta del servidor como una cadena de texto en formato JSON.
* res.json() espera un objeto JavaScript, no una cadena de texto. Si pasas una cadena directamente a res.json(), no se comportará como esperas, y podría causar errores o un comportamiento inesperado.

 **Necesidad de Parsear el JSON**:

* Antes de pasar la respuesta al cliente, necesitas convertir la cadena JSON en un objeto JavaScript utilizando JSON.parse().

app.use("/", router);

**Estructura de la Ruta**: Las rutas definidas en el enrutador (/airports, /flights, etc.) se unirán a la ruta base. Como montaste el enrutador en "/", las rutas se mantendrán tal como se definieron en el enrutador.

Si cambias la ruta base a algo como "/api", entonces todas las rutas en el enrutador estarán bajo /api, por ejemplo, /api/airports, /api/flights, etc.

**Frontend**

**SearchAutocomplete Material**

**npm install @mui/material @emotion/react @emotion/styled**

*  @mui/material: Este es el paquete principal de Material-UI, que incluye todos los componentes de UI (botones, tarjetas, inputs, etc.).
* @emotion/react y @emotion/styled: Estas bibliotecas son necesarias para el sistema de estilo de Material-UI.

**npm install @mui/icons-material**

Esto te permitirá utilizar los iconos predefinidos de Material Design en tus componentes.

**npm install axios**