DOCUMENTACIÓN

Análisis y primeras pruebas APIs.

DOCUMENTACIÓN APIs

Índice.

DOCUMENTACIÓN APIs

---

1. Reservas.

1.1. PUT /reservas/realizar-reserva.

- **Descripción:** Este endpoint permite a un cliente realizar una reserva de vuelos. La reserva incluye un vuelo de ida y un vuelo de vuelta, junto con la asignación de asientos a los pasajeros. El sistema verifica que los datos sean válidos, realiza la asignación de asientos y confirma la reserva. También gestiona errores si los datos proporcionados son incorrectos o si ocurre algún problema en el servidor.
- **Método HTTP**: POST.
- **Autenticación:** Es obligatorio que el cliente esté autenticado mediante un token JWT.
- **Parámetros de entrada:**

```
{

  "codigo_vuelo_ida": "UX23",
  "codigo_vuelo_vuelta": "UX24",
  "pasajeros": [
   {
    "nombre": "Juan",
    "apellidos": "Pérez García",
    "email": "juan@example.com",
    "telefono": "612345678",
    "nif": "12345678A",
    "ida": {
     "fila": "12",
     "columna": "A",
     "codigo_asiento": "12A",
     "precio": 150
    },
    "vuelta": {
     "fila": "10",
     "columna": "B",
     "codigo_asiento": "10B",
     "precio": 140
    }
   }
  ]
}
```

Respuestas posibles (código HTTP):

- **201:** Si la reserva se realiza con éxito.

```
{
  "message": "Reserva realizada con éxito",
```

```
  "reserva": {
    "success": true,
    "reservaId": 26
  }
}
```

- **400 :** Si falta algún dato necesario o los datos son incorrectos.
- **500:** Si ocurre un error en el servidor.

1.2. PUT /reservas/realizar-reserva-aleatoria.

- **Descripción:** Este endpoint permite a un cliente autenticado realizar una reserva para uno o varios pasajeros en un vuelo de ida y vuelta. Se asignan de manera automática y aleatoria los asientos disponibles para cada pasajero tanto en el trayecto de ida como en el de vuelta.

- **Método HTTP:** PUT**.**

- **Autenticación:** Es necesario que el cliente esté autenticado mediante un token JWT. El identificador del cliente se extrae del token y se utiliza para asociar la reserva al usuario correspondiente.

- **Parámetros de entrada:** El cuerpo de la solicitud en formato JSON.

```
{

 "codigo_vuelo_ida": "UX23",
 "codigo_vuelo_vuelta": "UX24",
 "pasajeros": [
  {
   "nombre": "Juan",
   "apellidos": "Pérez García",
   "email": "juan@example.com",
   "telefono": "612345678",
   "nif": "12345678A",
   "ida": {
    "fila": "12",
    "columna": "A",
    "codigo_asiento": "12A",
    "precio": 150
   },
   "vuelta": {
    "fila": "10",
    "columna": "B",
    "codigo_asiento": "10B",
    "precio": 140
   }
  }
 ]
```

```
}
```

Respuestas posibles (código HTTP):

- **201**: Reserva realizada correctamente. Se devuelve un mensaje de éxito junto con los datos de la reserva creada.

```
{
  "message": "Reserva realizada con éxito",
  "reserva": {
    "success": true,
    "reservaId": 26
  }
}
```

- **500:** Error inesperado durante la ejecución del proceso de reserva.

1.3. GET /reservas/mis-reservas

- **Descripción:** Esta API permite obtener todas las reservas realizadas por un cliente autenticado. Cada reserva contiene la información detallada de los billetes asociados, incluyendo los datos del vuelo, asiento y pasajero correspondiente.
- **Método HTTP:** GET.
- **Ruta:** /reservas/mis-reservas.
- **Autenticación:** Es obligatorio que el cliente esté autenticado mediante un token JWT.[1]
- **Parámetros de entrada:** Ninguno.
- **Estructura de la respuesta:** La respuesta contiene un listado de reservas. Cada reserva incluye:

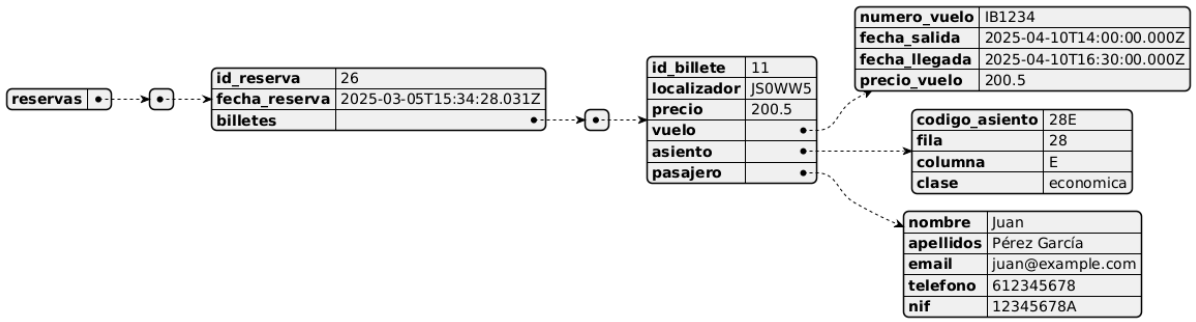
  Ejemplo de respuesta exitosa (200 OK):

```
{
 "reservas": [
  {
   "id_reserva": 26,
   "fecha_reserva": "2025-03-05T15:34:28.031Z",
   "billetes": [
    {
     "id_billete": 11,
     "localizador": "JS0WW5",
     "precio": "200.5",
     "vuelo": {
      "numero_vuelo": "IB1234",
      "fecha_salida": "2025-04-10T14:00:00.000Z",
```

[1] El identificador del cliente (id_cliente) se extrae del token para poder filtrar las reservas que le pertenecen.

```
        "fecha_llegada": "2025-04-10T16:30:00.000Z",
        "precio_vuelo": 200.5
       },
       "asiento": {
        "codigo_asiento": "28E",
        "fila": 28,
        "columna": "E",
        "clase": "economica"
       },
       "pasajero": {
        "nombre": "Juan",
        "apellidos": "Pérez García",
        "email": "juan@example.com",
        "telefono": "612345678",
        "nif": "12345678A"
       }
      }
     ]
    }
  ]
}
```


DIAGRAMA JSON



Respuestas posibles (código HTTP).

- **200:** Éxito. Devuelve la lista de reservas del cliente, mencionada anteriormente.
- **400:** Error. No se proporcionó el identificador del cliente.
- **404:** No se encontraron reservas para el cliente indicado.
- **500:** Error interno del servidor al obtener las reservas.

1.4. DELETE /reservas/:id_reserva.

- **Descripción:** Esta API permite a un cliente autenticado eliminar todas sus reservas. Borra todos billetes a los que está asociado la reserva. La eliminación solo se realiza si la reserva existe y pertenece al cliente que la solicita.

- **Método:** DELETE**.**
- **Ruta:** /reservas/:id_reserva
- **Autenticación:** Requiere autenticación mediante token JWT. El identificador del cliente (id_cliente) se extrae del token (req.user.sub) para validar la propiedad de la reserva.
- **Parámetros de entrada.**
    - *id_reserva:* Identificador único de la reserva a eliminar.
    - *id_cliente:* Token (JWT) Se obtiene del token de autenticación.

Respuestas posibles (código HTTP).

- **200:** Reserva eliminada correctamente.

```
{
  "message": "Reserva eliminada exitosamente."
}
```

- **400:** Faltan datos requeridos (cliente o reserva).
- **404:** No se encontró la reserva o no pertenece al cliente.
- **500:** Error interno del servidor.

1.5. DELETE reservas/:id_reserva/billetes/:id_billete.

- **Descripción:** Este endpoint permite a un cliente autenticado eliminar un billete específico dentro de una de sus reservas. La operación solo se realiza si el billete pertenece a la reserva indicada y si dicha reserva pertenece al cliente que realiza la solicitud.
- **Método HTTP:** DELETE.
- **Ruta:** /reservas/:id_reserva/billetes/:id_billete
- **Autenticación** Requiere autenticación mediante token JWT. El identificador del cliente (id_cliente) se obtiene del token.

- **Parámetros de entrada**
    - id_reserva: Identificador único de la reserva.
    - id_billete: Identificador único del billete a eliminar.
    - id_cliente -> Token (JWT): Se extrae del token de autenticación.

Respuestas posibles (código HTTP):.

- **200:**

```
{
  "message": "Billete eliminado exitosamente."
}
```

- **400:** Datos insuficientes.
- **404:** No se encontró el billete solicitado en esta reserva.
- **500:** Error interno del servidor.

DOCUMENTACIÓN APIs

---

2. Aeropuertos.

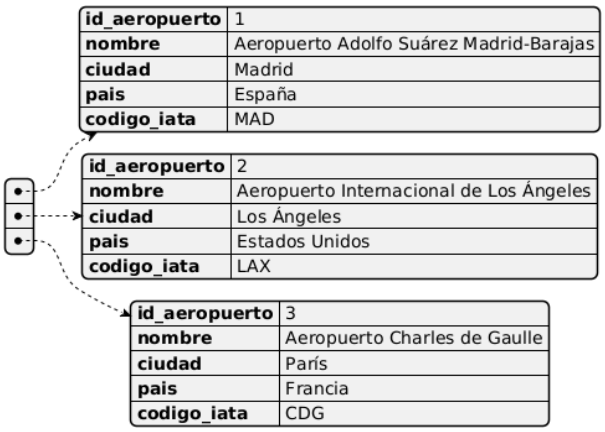
2.1. GET /aeropuertos.

- **Descripción:** esta API permite obtener la lista de todos los aeropuertos disponibles.
- **Método HTTP:** GET.
- **Endpoint:** /aeropuertos.
- **Parámetros de entrada:** ninguno.

Respuestas posibles (Código HTTP).

- **200**: devolverá una lista de aeropuertos.

```
[
  {
    "id_aeropuerto": 1,
    "nombre": "Aeropuerto Adolfo Suárez Madrid-Barajas",
    "ciudad": "Madrid",
    "pais": "España",
    "codigo_iata": "MAD"
  },
  ……
]
```



- **500:** Error al obtener los aeropuertos.

3. Asientos.

3.1. GET /asientos/vuelo/:numero_vuelo.

- **Descripción:** Esta API permite obtener la distribución de los asientos de un vuelo específico, dado el número de vuelo. La respuesta incluye la información detallada de los asientos en formato estructurado, con la fila, columna, clase y el estado de cada asiento (*disponible o reservado*).
- **Método HTTP:** GET.

DOCUMENTACIÓN APIs

- **Endpoint:** /vuelos/:numero_vuelo/asientos.
- **Parámetros de entrada:** numero_vuelo (requerido).

Respuestas posibles (código HTTP).

- **200:** La API devolverá la distribución de asientos del vuelo solicitado .


```
{
  "avion": "Airbus A320",
  "distribucion_asientos": [
    {
      "fila": 1,
      "asientos": [
        {
          "codigo_asiento": "1A",
          "fila": 1,
          "columna": "A",
          "clase": "Business",
          "estado": "reservado"
        },
        {
          "codigo_asiento": "1B",
          "fila": 1,
          "columna": "B",
          "clase": "Business",
          "estado": "disponible"
        },
        … ]
    },
    {
      "fila": 2,
      "asientos": [
        {
          "codigo_asiento": "2A",
          "fila": 2,
          "columna": "A",
          "clase": "Business",
          "estado": "disponible"
        },
        … ]
    },
    ...
  ]
}
```


- **500:** Problema al devolver los asientos.

DOCUMENTACIÓN APIs

---

4. Clientes.

4.1. GET /clientes.

- **Método:** GET
- **Descripción:** Obtiene los datos del cliente autenticado por su ID.
- **Ruta:** /clientes/perfil-cliente
- **Respuesta exitosa (200):**

```
{
  "id_cliente": 1,
  "nombre": "Juan",
  "apellidos": "Pérez",
  "email": "juan@example.com",
  "telefono": "123456789",
  "nif": "12345678X",
  "nombre_usuario": "juanito”
}
```

Respuestas posibles (Código HTTP):

- **404:** Cliente no encontrado.
- **500:** Error al obtener el cliente.

4.2. POST /clientes.

- **Método:** POST
- **Descripción:** Registra un nuevo cliente en el sistema.
- **Ruta:** /clientes
- **Parámetros de entrada:** Cuerpo de la solicitud (JSON):

```
{
  "nombre": "Juan",
  "apellidos": "Pérez",
  "email": "juan@example.com",
  "telefono": "123456789",
  "nif": "12345678X",
  "contraseña": "miContraseña123",
  "nombre_usuario": "juanito”
}
```

Respuestas posibles (código HTTP).

- **201:**

```
{
  "message": "Cliente creado con éxito"
}
```

- **400:** Todos los campos son obligatorios.
- **500:** Error al crear el cliente.

DOCUMENTACIÓN APIs

4.3. POST /clientes/login.

- **Método:** POST
- **Descripción:** Inicia sesión de un cliente y envía un código de verificación al correo electrónico.
- **Ruta:** /clientes/login
- **Parámetros de entrada:** Cuerpo de la solicitud (JSON):

```
{
    "usuarioOEmail": "juanito",
    "contraseña": "miContraseña123"
}
```

Respuestas posibles (Código HTTP):

- **200:** Éxito al loguear al cliente.

```
{
     "message": "Inicio de sesión exitoso. Se ha enviado un código de verificación
    al correo.",
     "token": "jwt_token_aqui",
     "estaLogueado": true
}
```

- **401:** Credenciales incorrectas.
- **500:** Error al intentar iniciar sesión.

4.4. PUT /clientes.

- **Método:** PUT
- **Descripción:** Actualiza los datos de un cliente autenticado.
- **Ruta:** /clientes/actualizar-datos-cliente
- **Parámetros de entrada:** Cuerpo de la solicitud (JSON):

```
{
  "nombre": "Juan",
  "apellidos": "Pérez",
  "email": "juan@example.com",
  "telefono": "123456789",
  "contraseña": "nuevaContraseña123",
  "nombre_usuario": "juanito"
}
```

Respuestas posibles (código HTTP):

- **Respuesta exitosa (200):**

```
{
     "message": "Cliente actualizado con éxito"
}
```

- **Respuesta de error (404):** Cliente no encontrado.
- **Respuesta de error (500):** Error al actualizar el cliente.

4.5. DELETE /clientes.

- **Método:** DELETE
- **Descripción:** Elimina un cliente autenticado del sistema.
- **Ruta:** /clientes/:eliminar-cuenta

Respuestas posibles (código HTTP):

- **200:**

```
{
        "message": "Cliente eliminado con éxito"
}
```

- **404:** Cliente no encontrado.
- **500:** Error al eliminar el cliente.

5. Vuelos.

5.1. GET /vuelos/buscador-vuelos.

- **Descripción:** obtiene vuelos de ida y vuelta entre aeropuertos, filtrando por fecha y número de pasajeros.
- **Método HTTP**: GET.
- **Parámetros de entrada:**
  - codigo_origen (string): Código IATA del aeropuerto de origen (Ejemplo: "MAD").
  - codigo_destino (string): Código IATA del aeropuerto de destino (Ejemplo: "LAX").
  - fecha_ida (string): Fecha de salida en formato YYYY-MM-DD.
  - fecha_vuelta (string): Fecha de regreso en formato YYYY-MM-DD.
  - numero_pasajeros (integer): Número de pasajeros (por defecto 1 si no se envía).

Ejemplo en formato JSON

```
{
        "codigo_origen": "MAD",
        "codigo_destino": "LAX",
        "fecha_ida": "2025-06-15",
        "fecha_vuelta": "2025-06-30",
        "numero_pasajeros": 2
}
```

Respuestas posibles (código HTTP).

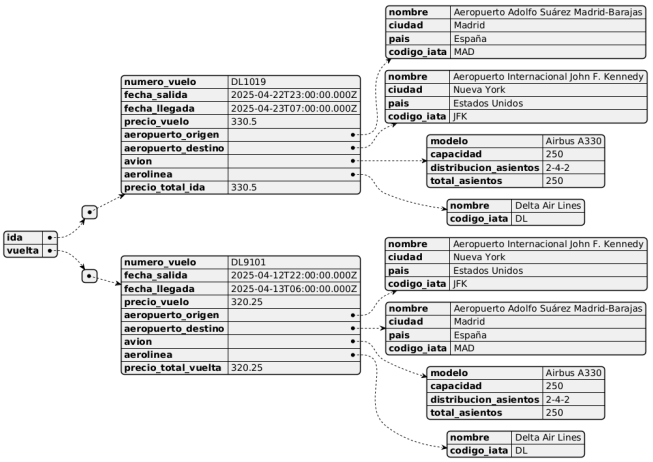
- **200**: La respuesta es un objeto con dos listas:
  - ida: Vuelos de ida disponibles.
  - vuelta: Vuelos de regreso disponibles (*solo si se indica fecha_vuelta*).

EJEMPLO DE RESPUESTA (JSON)


```
{
  "ida": [
    {
      "numero_vuelo": "DL1019",
      "fecha_salida": "2025-04-22T23:00:00.000Z",
      "fecha_llegada": "2025-04-23T07:00:00.000Z",
      "precio_vuelo": 330.5,
      "aeropuerto_origen": {
        "nombre": "Aeropuerto Adolfo Suárez Madrid-Barajas",
        "ciudad": "Madrid",
        "pais": "España",
        "codigo_iata": "MAD"
      },
      "aeropuerto_destino": {
        "nombre": "Aeropuerto Internacional John F. Kennedy",
        "ciudad": "Nueva York",
        "pais": "Estados Unidos",
        "codigo_iata": "JFK"
      },
      "avion": {
        "modelo": "Airbus A330",
        "capacidad": 250,
        "distribucion_asientos": "2-4-2",
        "total_asientos": 250
      },
      "aerolinea": {
        "nombre": "Delta Air Lines",
        "codigo_iata": "DL"
      },
      "precio_total_ida": 330.5
    }
  ],
  "vuelta": [
    {
      "numero_vuelo": "DL9101",
      "fecha_salida": "2025-04-12T22:00:00.000Z",
      "fecha_llegada": "2025-04-13T06:00:00.000Z",
      "precio_vuelo": 320.25,
      "aeropuerto_origen": {
        "nombre": "Aeropuerto Internacional John F. Kennedy",
        "ciudad": "Nueva York",
        "pais": "Estados Unidos",
        "codigo_iata": "JFK"
      },
      "aeropuerto_destino": {
        "nombre": "Aeropuerto Adolfo Suárez Madrid-Barajas",
        "ciudad": "Madrid",
```

```
            "pais": "España",
            "codigo_iata": "MAD"
          },
          "avion": {
            "modelo": "Airbus A330",
            "capacidad": 250,
            "distribucion_asientos": "2-4-2",
            "total_asientos": 250
          },
          "aerolinea": {
            "nombre": "Delta Air Lines",
            "codigo_iata": "DL"
          },
          "precio_total_vuelta": 320.25
        }
      ]
}
```


DIAGRAMA JSON



- **500:** Si ocurre un error al procesar la solicitud, la respuesta incluirá un mensaje de error con los detalles del fallo.

5.6. POST /vuelos.

- **Descripción:** Se crea un nuevo vuelo en el sistema con la información proporcionada.
- **Método HTTP:** POST.
- **Ruta:** /vuelos.
- **Parámetros de entrada:**
  - numero_vuelo (string) - Número identificador del vuelo.
  - fecha_salida (string, formato ISO 8601) - Fecha y hora de salida del vuelo.
  - fecha_llegada (string, formato ISO 8601) - Fecha y hora de llegada del vuelo.
  - id_aeropuerto_origen (integer) - ID del aeropuerto de origen.
  - id_aeropuerto_destino (integer) - ID del aeropuerto de destino.
  - id_avion (integer) - ID del avión asignado al vuelo.
  - id_aerolinea (integer) - ID de la aerolínea operadora.
  - precio_vuelo (float) - Precio del vuelo.

Respuestas posibles (código HTTP):

- **201 (Created):** Vuelo creado exitosamente.
- **400 (Bad Request):** Faltan parámetros obligatorios o los datos tienen un formato inválido.
- **500 (Internal Server Error)**: Error en el servidor al procesar la solicitud.

5.7. GET /vuelos.

- **Descripción:** Obtiene la lista de todos los vuelos disponibles en la base de datos.
- **Método HTTP:** GET**.**
- **Ruta:** /vuelos.
- **Parámetros de entrada:** Ninguno.

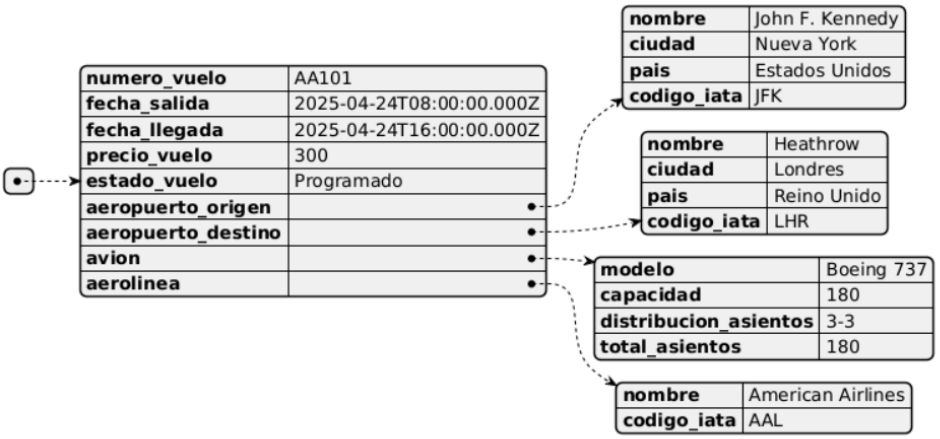
Respuestas posibles (código HTTP):

- **200:** Éxito al obtener los vuelos. Devuelve un listado con los vuelos disponibles.


```
[
  {
    "numero_vuelo": "AA101",
    "fecha_salida": "2025-04-24T08:00:00.000Z",
    "fecha_llegada": "2025-04-24T16:00:00.000Z",
    "precio_vuelo": 300,
    "estado_vuelo": "Programado",
    "aeropuerto_origen": {
      "nombre": "John F. Kennedy",
      "ciudad": "Nueva York",
      "pais": "Estados Unidos",
      "codigo_iata": "JFK"
    },
    "aeropuerto_destino": {
      "nombre": "Heathrow",
      "ciudad": "Londres",
      "pais": "Reino Unido",
      "codigo_iata": "LHR"
    },
    "avion": {
      "modelo": "Boeing 737",
      "capacidad": 180,
      "distribucion_asientos": "3-3",
      "total_asientos": 180
    },
    "aerolinea": {
```

```
        "nombre": "American Airlines",
        "codigo_iata": "AAL"
    }
 },
……..
]
```


DIAGRAMA JSON DE CADA ELEMENTO DE LA LISTA.



- **500**: Error al obtener los vuelos.

5.8. PUT /vuelos/modificar-estado-vuelo.

- **Descripción:** Modifica el estado de un vuelo en la base de datos. Únicamente se acepta el valor "C" para el estado del vuelo, lo que lo marcará como "Cancelado".
- **Método HTTP:** PUT.
- **Ruta:** /vuelos/modificar-estado-vuelo.
- **Parámetros de entrada:**
  - **codigo_vuelo** *(string)*: Código único del vuelo a modificar.
  - **estado_vuelo** *(string)*: Debe ser "C" para cancelar el vuelo.

Respuestas posibles (código HTTP):

- **200:** Vuelo modificado con éxito.


```
{
    "mensaje": "Vuelo modificado con éxito"
}
```


- **400:** Faltan parámetros en la solicitud o el estado_vuelo es distinto “C”.
- **404:** Vuelo no encontrado.
- **500**: Error interno del servidor.

DOCUMENTACIÓN APIs

---

6. Billetes.

6.1. GET /billetes/obtener-info-vuelo.

- **Descripción:** esta API permite a un usuario consultar los detalles de un billete proporcionando el localizador y el apellido del pasajero. Este endpoint es útil para recuperar información de una reserva sin necesidad de estar autenticado previamente, como en casos de consultas previas al check-in o recuperación de billetes.
- **Método HTTP:** GET.
- **Ruta:** /billetes/obtener-info-vuelo.
- **Parámetros de entrada:**

  localizador (string): Código único asociado al billete.
  apellido (string): Apellido(s) del pasajero.

Respuestas posibles (código HTTP):

- **200:** Billete encontrado.


```
{
  "success": true,
  "data": {
    "billete": {
      "localizador": "XFFOLB",
      "precio": "210.75"
    },
    "reserva": {
      "fecha": "2025-03-05T15:34:28.031Z"
    },
    "pasajero": {
      "nombre": "Juan",
      "apellidos": "Pérez García",
      "email": "juan@example.com",
      "telefono": "612345678",
      "nif": "12345678A"
    },
    "vuelo": {
      "numero": "IB4321",
      "fecha_salida": "2025-04-20T12:00:00.000Z",
      "fecha_llegada": "2025-04-20T20:30:00.000Z",
      "precio_vuelo": 210.75,
      "aerolinea": "Iberia",
      "avion": "Airbus A320",
      "origen": {
        "aeropuerto": "Aeropuerto Internacional de Los Ángeles",
        "ciudad": "Los Ángeles",
        "pais": "Estados Unidos"
      },
      "destino": {
```

```
        "aeropuerto": "Aeropuerto Adolfo Suárez Madrid-Barajas",
        "ciudad": "Madrid",
        "pais": "España"
       }
      },
      "asiento": {
       "codigo": "23B",
       "clase": "economica",
       "estado": "reservado"
      }
     }
    }
```


- **400:** Faltan datos.
- **404:** Billete no encontrado.
- **500**: Error del servidor.

7. Aerolíneas.

7.1. GET /aerolineas.

- **Descripción:** Obtiene la lista de todas las aerolíneas registradas en el sistema.
- **Método HTTP:** GET.
- **Ruta:** /aerolineas.
- **Autenticación:** Es obligatorio que el usuario esté autenticado con un token JWT y sea administrador.
- **Parámetros de entrada:** No se requieren parámetros en la solicitud.

Respuestas posibles (código HTTP).

- **200:**


```
[
  {
    "id_aerolinea": 1,
    "nombre": "American Airlines",
    "codigo_iata": "AAL"
  },
]
```


- **500**: Error del servidor.

8. Aviones.

8.1. GET /aviones

- **Descripción:** Obtiene la lista de todos los aviones registrados en el sistema.
- **Método HTTP:** GET.
- **Ruta:** /aviones.

- **Autenticación:** Es obligatorio que el usuario esté autenticado con un token JWT y sea administrador.
- **Parámetros de entrada:** No se requieren parámetros en la solicitud.

Respuestas posibles (código HTTP).

- **200:** Éxito al obtener los vuelos.

```
[
  {
    "id_avion": 1,
    "modelo": "Boeing 737",
    "capacidad": 180,
    "distribucion_asientos": "3-3",
    "total_asientos": 180,
    "id_aerolinea": 1
  },
 ]
```

- **500**: Error del servidor.

9. Anexos.

9.1. Código de respuesta de encabezado.

9.1. 2xx (Success).

- 200 (*OK)*: La solicitud fue exitosa y el servidor devolvió la respuesta esperada.
- 201 (*Created*): La solicitud se completó y se creó un nuevo recurso.

9.2. 4xx (Client Error).

- 400 (*Bad Request*): La solicitud tiene un error de sintaxis o no puede ser procesada.
- 401 (*Unauthorized*): Falta autenticación o es inválida.
- 402 (*Payment Required*): Reservado para futuros usos relacionados con pagos.
- 403 (*Forbidden*): El servidor deniega el acceso aunque la autenticación sea válida.
- 404 (*Not Found*): El recurso solicitado no existe en el servidor.

9.3. 5xx (Server Error).

- 500 (*Internal Server Error*): Error general del servidor sin una causa específica.
- 502 (*Bad Gateway*): El servidor recibió una respuesta inválida de otro servidor.

- 503 (*Service Unavailable*): El servidor está sobrecargado o en mantenimiento.
- 504 (*Gateway Timeout*): El servidor no recibió una respuesta a tiempo desde otro servidor.

9.2. URL base (despliegue).

- URL_BASE_DESPLIGUE = https://nodejs-mysql-mdzc.onrender.com

9.3. Token y autenticación con JWT.

- **Autenticación con JWT.**

  La autenticación basada en JSON Web Tokens (JWT) es un esquema de seguridad que utiliza tokens para validar la identidad de un usuario. Un JWT es un token firmado digitalmente que contiene información sobre el usuario y puede ser usado para autenticar solicitudes posteriores sin necesidad de enviar credenciales repetidas.

  El token encriptado contiene esta información (*id* del cliente, el *email*, el *nombre de usuario* y si es *administrador*) almacenada en un objeto (payload) que posteriormente será encriptado.

```
const payload = {
  sub: cliente.id_cliente,
  email: cliente.email,
  name: cliente.nombre_usuario,
  is_admin: !!cliente.es_admin,
};
```

  Funciones importantes *(Middleware[2])* para manejar las autenticaciones.

  - **verificarToken**: Verifica que el token JWT esté presente y sea válido; si es válido, lo decodifica y lo asocia al objeto req.user, permitiendo el acceso a la siguiente función.

```
/**********************RUTA RESERVA CONTROLLER*********************************/
router.put('/reservas/realizar-reserva', verificarToken, reservaController.realizarReserva);
router.put('/reservas/realizar-reserva-aleatoria', verificarToken, reservaController.realizarReservaConAsignacionAleatoria)
router.get('/reservas/mis-reservas', verificarToken, reservaController.obtenerReservaCliente);
router.delete('/reservas/:id_reserva', verificarToken, reservaController.eliminarReservaCliente);
router.delete('/reservas/:id_reserva/billetes/:id_billete', verificarToken, reservaController.eliminarBilleteDeReserva);
```

  - **verificarAdmin**: Verifica si el usuario tiene permisos de administrador, y si es así, permite continuar, de lo contrario, deniega el acceso con un error 403.

```
/*Solo el administrador puede crear los vuelos*/
router.post('/vuelos', verificarToken, verificarAdmin, vueloController.createVuelo);
```

[2] Los middleware actúan como **puentes o filtros** entre el cliente (usuario que hace la solicitud) y el servidor (donde se maneja la lógica de negocio).

DOCUMENTACIÓN APIs

DIAGRAMA CON LA LÓGICA DEL LOGIN Y LA AUTENTICACIÓN