INF-343 SISTEMAS DISTRIBUIDOS

TAREA N°1: “Desarrollo de una aplicación distribuida simple de reserva aérea usando servicios web”

16 de agosto de 2023

ANTECEDENTES GENERALES

Objetivos:

- Aprender a programar con el lenguaje de programación concurrente Go, desarrollado por Google (https://go.dev/).
- Desarrollar una aplicación con servicios Web basados en RESTful APIs, programando con lenguaje Go y usando Gin Web Framework. Se usará como base de datos (NoSQL) MongoDB para la persistencia de los datos.

Fechas importantes:

- **Ayudantía por Zoom:**
- **Fecha de entrega de la tarea**: 13 de septiembre de 2023

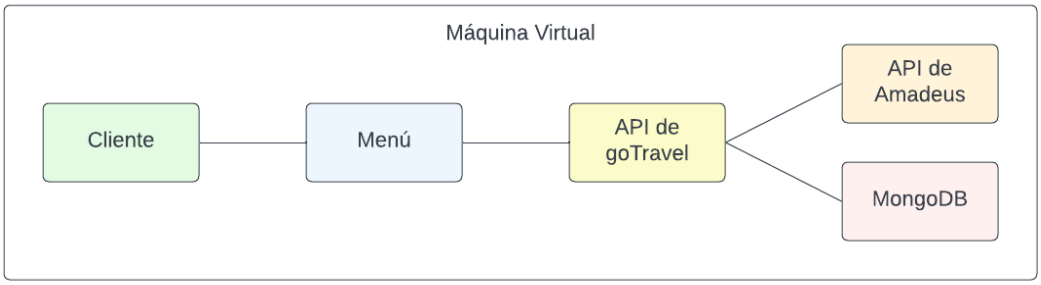
ESPECIFICACIÓN DE LA APLICACIÓN

Amadeus es un proveedor que muchas aerolíneas —como British Airways, KLM, Lufthansa, Qatar Airways, entre otras—, utilizan para permitir que los clientes gestionen sus reservas, que el personal del aeropuerto despache equipajes o que las agencias vendan pasajes actuando como intermediarios. En esta tarea nos enfocaremos en el alcance que tienen las agencias, la que básicamente es vender pasajes de distintas aerolíneas para que el cliente cuente con diversas opciones.

La tarea consiste en desarrollar un sistema para que una agencia (goTravel) realice una reserva para uno o varios pasajeros, teniendo ciertas limitaciones que se detallan en este mismo enunciado. La agencia tendrá disponible dos opciones:

1. Crear una reserva.
2. Obtener los datos de una reserva.

Para ambas opciones, se compartirá la siguiente arquitectura:



Cada componente se detalla a continuación:

1. Cliente: Es la terminal del SO donde se iniciará la interacción con un menú de opciones.
2. Menú: Es el despliegue de opciones para que el cliente realice distintas acciones. También se obtiene el resultado de las opciones seleccionadas.
3. API de goTravel: Es la API que se comunicará con la API de Amadeus y MongoDB, tanto para consultar información como para persistir datos.
4. API de Amadeus: Es la API propia de Amadeus, la cual nos entregará la información para crear las reservas.
5. MongoDB: Es nuestro sistema de base de datos para persistir las reservas.

DESARROLLO

API de Amadeus

Registro

Antes de consultar cualquier API de Amadeus, debemos registrarnos, iniciar sesión y luego crear una aplicación en su sitio *web* de desarrolladores: https://developers.amadeus.com/. Así obtendremos un ID de cliente y una clave secreta.

Autorización

Una vez que se haya obtenido un ID de cliente (*client_id*) y una clave secreta (*client_secret*), debemos obtener un token de autorización, para que así tengamos el permiso de consultar los *endpoints* de Amadeus. Observe el siguiente cURL:

```
curl --location 'https://test.api.amadeus.com/v1/security/oauth2/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data
'grant_type=client_credentials&client_id={client_id}&client_secret={client_sec
ret}'
```

Una vez ejecutado, obtendremos un *token* (`access_token`):

```
{
    "type": "amadeusOAuth2Token",
    "username": "hleytondiaz@gmail.com",
    "application_name": "Example",
    "client_id": "rHD7EuPExXjGT0SF53UACYiQlZsF1nkX",
    "token_type": "Bearer",
    "access_token": "DsTVa7saplNUwdWSiNU8WgooGDpr",
    "expires_in": 1799,
    "state": "approved",
    "scope": ""
}
```


Búsqueda

Para crear una reserva, primero debemos buscar los vuelos que nos interesan, debiendo tener en cuenta los siguientes atributos:

Atributos	**Ejemplo**	**Descripción**
originLocationCode	ARI	Aeropuerto de origen
destinationLocationCode	SCL	Aeropuerto de destino
departureDate	2023-12-02	Fecha de inicio
returnDate**	2023-12-04	Fecha de regreso (opcional)
adults	1	Cantidad de adultos
includedAirlineCodes*	H2,LA,JA	Aerolíneas
nonStop*	true	Opción de incluir escalas
currencyCode*	CLP	Divisa
travelClass*	ECONOMY	Clase tarifaria

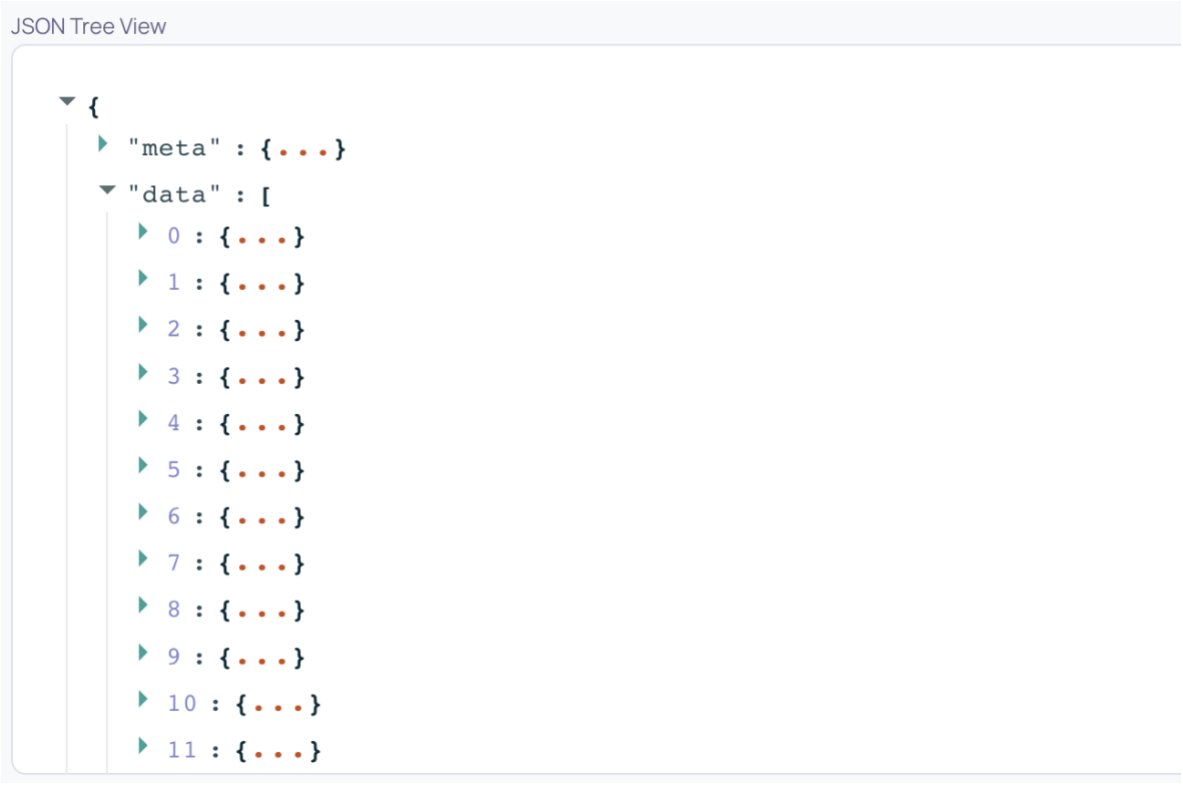
***** Por simplicidad, estos atributos siempre tendrán los mismos valores del ejemplo para cualquier búsqueda.

****** Por simplicidad, el atributo `returnDate` será descartado, por lo que no consideraremos vuelos *round trip*, solo *one way*.

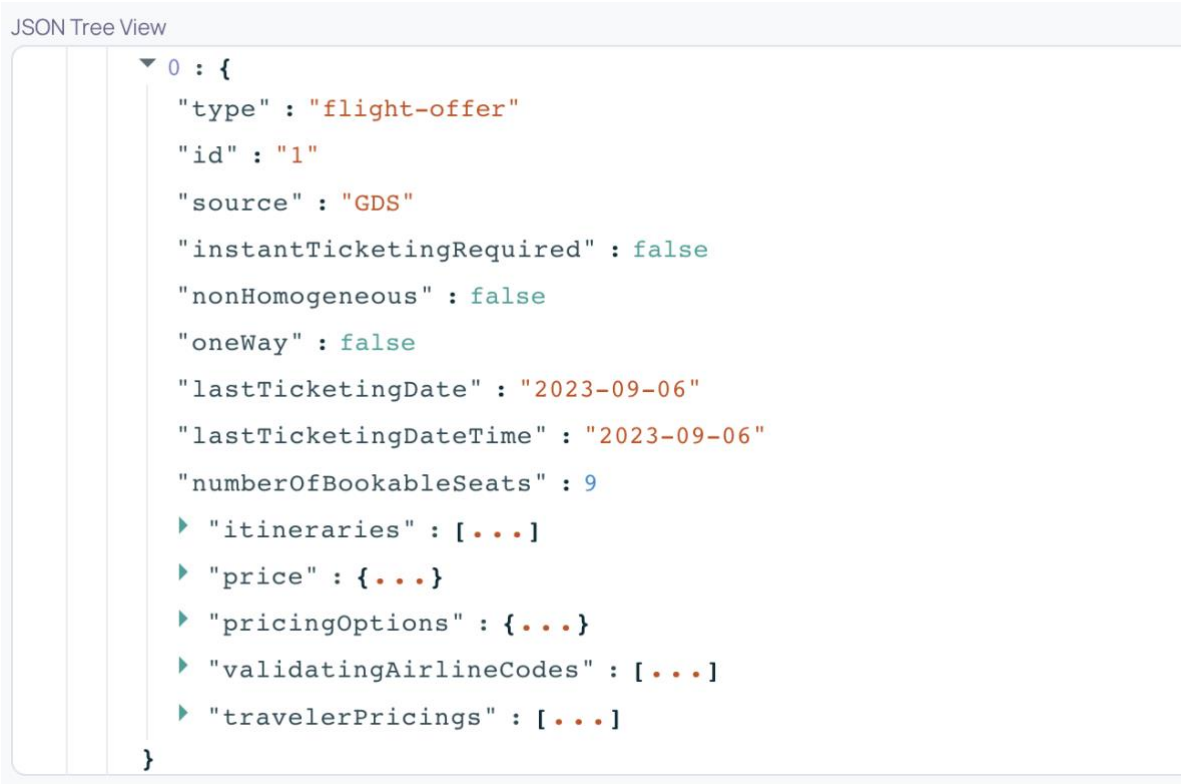
Luego, con el siguiente cURL podemos obtener los vuelos disponibles:

```
curl --location 'https://test.api.amadeus.com/v2/shopping/flight-offers?originLocationCode=DXB&destinationLocationCode=BKK&departureDate=2023-12-02&returnDate=2023-12-04&adults=1&includedAirlineCodes=EK&nonStop=true&currencyCode=CLP&travelClass=ECONOMY' \
--header 'Authorization: Bearer 2SOp462CNvkdsH5wb5DuGPY7Udfz'
```

La respuesta es la siguiente, donde cada índice de *data* corresponde a un resultado (ya sea de solo ida o ida y vuelta):



Un elemento de *data*, tendrá la siguiente estructura:



La sección que más nos interesa es `itineraries`, `price` y `travelerPricings`.

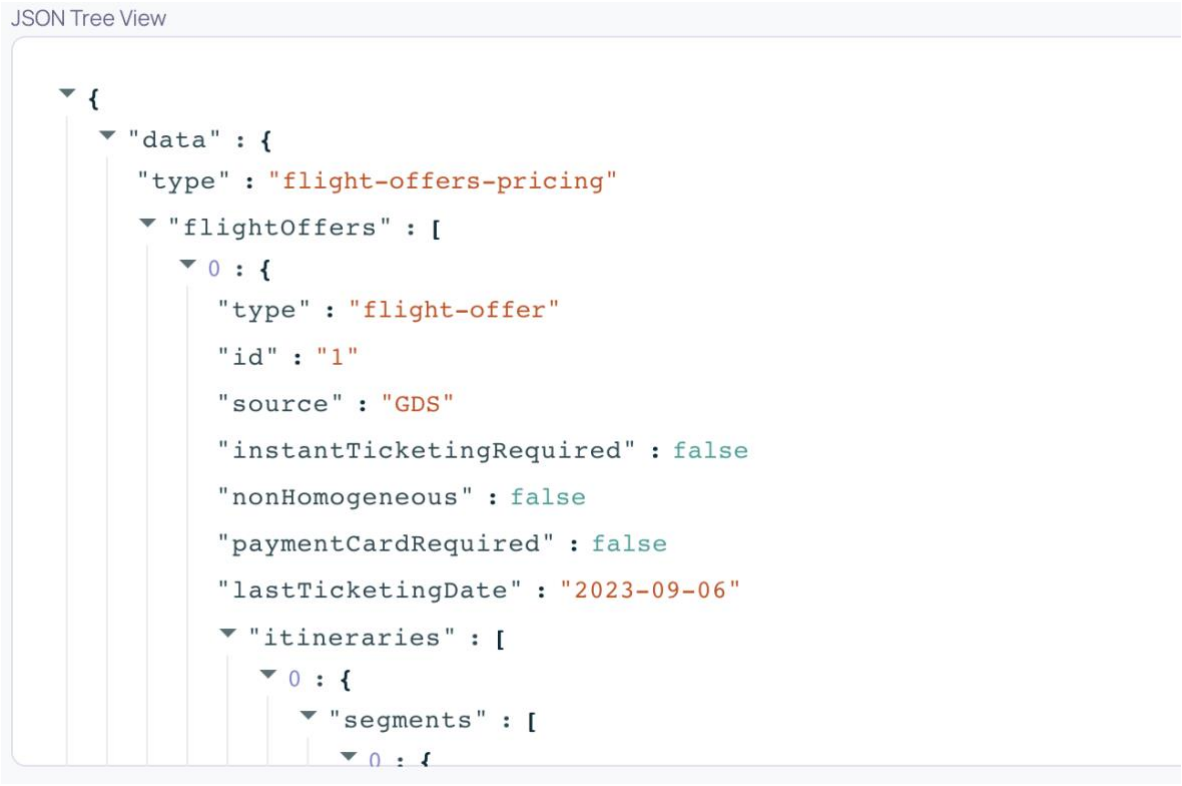
Fuente: https://developers.amadeus.com/self-service/category/flights/api-doc/flight-offers-search

Precios definitivos

Aunque Amadeus nos despliegue inicialmente los precios, no es seguro que al momento de decidirnos por algún resultado de búsqueda aún sean los mismos, así que Amadeus dispone del siguiente cURL para entregarnos los precios definitivos:

```
curl        --location        'https://test.api.amadeus.com/v1/shopping/flight-
offers/pricing' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer 2SOp462CNvkdsH5wb5DuGPY7Udfz' \
--data '{
  "data": {
    "type": "flight-offers-pricing",
    "flightOffers": [...]
  }
}'
```

Note que `flightOffers` ha sido minimizado porque corresponde a la misma *data* que incluye el cURL de búsqueda. La respuesta que entrega este cURL es similar al de buscar:



Fuente: https://developers.amadeus.com/self-service/category/flights/api-doc/flight-offers-price

Crear reserva

Cuando el precio ya está confirmado, solo resta ejecutar el cURL que crea la reserva. Aunque primero debemos tener en cuenta la información básica de uno o varios pasajeros, estos son:

1. Fecha de nacimiento.
2. Nombre.
3. Apellido.
4. Sexo.
5. Correo electrónico.
6. Número de teléfono.

Ya con los datos listos, se ejecuta el siguiente cURL:

```
curl --location 'https://test.api.amadeus.com/v1/booking/flight-orders' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer DsTVa7saplNUwdWSiNU8WgooGDpr' \
--data-raw '{
    "data": {
        "type": "flight-order",
        "flightOffers": [...],
        "travelers": [
            {
                "id": "1",
                "dateOfBirth": "1982-01-16",
                "name": {
                    "firstName": "JORGE",
                    "lastName": "GONZALEZ"
                },
                "gender": "MALE",
                "contact": {
                    "emailAddress": "jorge.gonzalez833@telefonica.es",
                    "phones": [
                        {
                            "deviceType": "MOBILE",
                            "countryCallingCode": "34",
                            "number": "480080076"
                        }
                    ]
                },
                "documents": [...]
            }
        ],
        "remarks": {...},
        "ticketingAgreement": {...},
        "contacts": [...]
    }
}'
```

El cual nos entregará la siguiente estructura:

```
{
    "data": {
        "type": "flight-order",
        "id": "eJzTd9cPtbSIdDYFAAryAj0%3D",
        "queuingOfficeId": "NCE4D31SB",
        "associatedRecords": [
            {
                "reference": "U98YC5",
                "creationDate": "2023-08-16T03:20:00.000",
                "originSystemCode": "GDS",
                "flightOfferId": "1"
            }
        ],
        "flightOffers": [...],
        "travelers": [...]
    ...
    }
}
```


El dato más importante es el `id`, ya que corresponde al número identificador de la reserva creada.

Fuente: https://developers.amadeus.com/self-service/category/flights/api-doc/flight-create-orders

Obtener una reserva
Finalmente, si queremos recuperar los datos de una reserva ya creada, podemos ejecutar el siguiente cURL:

```
curl --location 'https://test.api.amadeus.com/v1/booking/flight-orders/eJzTd9cPtbSIdDYFAAryAj0%3D' \
--header 'Authorization: Bearer DsTVa7saplNUwdWSiNU8WgooGDpr'
```

La respuesta que obtendremos será similar a la que obtuvimos cuando creamos la reserva.

Fuente: https://developers.amadeus.com/self-service/category/flights/api-doc/flight-order-management

API de goTravel

La API de goTravel es la encargada de comunicarse entre el cliente, la API de Amadeus y MongoDB. Los cuatro *endpoints* se detallan a continuación:

Endpoint	127.0.0.1:5000/api/search
Método	GET
Parámetros	Los mismos que recibe /shopping/flight-offers
Respuesta (200)	La misma respuesta que entrega /shopping/flight-offers
Respuesta (4XX)	{“message”: “Hubo un error al realizar la búsqueda”}
Respuesta (5XX)	{“message”: “Hubo un error al realizar la búsqueda”}

Endpoint	127.0.0.1:5000/api/pricing
Método	POST
Body	El mismo que recibe /shopping/flight-offers/pricing
Respuesta (200)	La misma respuesta que entrega /shopping/flight-offers/pricing
Respuesta (4XX)	`{“message”: “Hubo un error al retornar los precios”}`
Respuesta (5XX)	`{“message”: “Hubo un error al retornar los precios”}`

Endpoint	127.0.0.1:5000/api/booking
Método	POST
Body	El mismo que recibe /booking/flight-orders
Respuesta (200)	La misma respuesta que entrega /booking/flight-orders
Respuesta (4XX)	`{“message”: “Hubo un error al crear la reserva”}`
Respuesta (5XX)	`{“message”: “Hubo un error al crear la reserva”}`

Endpoint	127.0.0.1:5000/api/booking
Método	GET
Parámetro	ID de la reserva
Respuesta (200)	La misma respuesta que entrega /booking/flight-orders/{id}
Respuesta (4XX)	`{“message”: “Hubo un error al recuperar la reserva”}`
Respuesta (5XX)	`{“message”: “Hubo un error al recuperar la reserva”}`

MongoDB

El uso de MongoDB tiene que ver con la tarea de persistir cada reserva que se crea. Las reservas serán registradas en una colección llamada *reservations* y la estructura que contendrán será la misma que la que retorna `/api/booking` con el método POST.

Menú

Cuando desde el cliente se ingresé al menú, se obtendrán las siguientes opciones:

1. Realizar búsqueda.
2. Obtener reserva.
3. Salir.

Si la opción 1 es seleccionada, se deberán ingresar los siguientes datos:

1. Aeropuerto de origen (en formato IATA).
2. Aeropuerto de destino (en formato IATA).
3. Fecha de salida (en ISO 8601 YYYY-MM-DD).
4. Cantidad de adultos.

Luego, se desplegarán todos los vuelos disponibles, pero incluyendo los siguientes atributos para cada uno:

1. Número de vuelo.
2. Fecha y hora de salida.
3. Fecha y hora de llegada.
4. Avión.
5. Precio total.

Posteriormente hay dos opciones: seleccionar una búsqueda y crear una reserva o realizar una nueva búsqueda. Si la primera opción es seleccionada, se mostrará el precio definitivo; después, se solicitará la cantidad de pasajeros y luego los siguientes datos para cada uno:

1. Fecha de nacimiento (en ISO 8601 YYYY-MM-DD).
2. Nombre.
3. Apellido.
4. Sexo (MALE o FEMALE).
5. Correo electrónico.
6. Número de teléfono.

Cuando los datos personales de todos los pasajeros hayan sido ingresados, la reserva será creada y se mostrará el ID.

Cuando se quiera obtener una reserva desde la opción 2 del menú principal, bastará con pedir el ID de la reserva, para así desplegar los siguientes datos:

1. Número de vuelo.
2. Fecha y hora de salida.
3. Fecha y hora de llegada.
4. Avión.
5. Precio total.
6. Nombre y apellido de cada pasajero.

Observe los siguientes ejemplos, que se ilustran para diferentes casos:

```
Bievenido a goTravel!
1. Realizar búsqueda.
2. Obtener reserva.
3. Salir
Ingrese una opción: 1
Aeropuerto de origen: ARI
Aeropuerto de destino: SCL
Fecha de salida: 2023-12-02
Cantidad de adultos: 1
Se obtuvieron los siguientes resultados:
+-------+--------+----------------+-----------------+-------+--------------------+
| VUELO | NÚMERO | HORA DE SALIDA | HORA DE LLEGADA | AVIÓN | PRECIO TOTAL       |
+-------+--------+----------------+-----------------+-------+--------------------+
|     1 | LA103  | 11:26          | 12:21           | A320  |           55952.00 |
|     2 | LA107  | 17:59          | 18:54           | A321  |           61951.00 |
|     3 | H2207  | 16:06          | 17:00           | A320  |           71378.00 |
|     4 | H2201  | 12:11          | 13:06           | A320  |           71378.00 |
+-------+--------+----------------+-----------------+-------+--------------------+
```

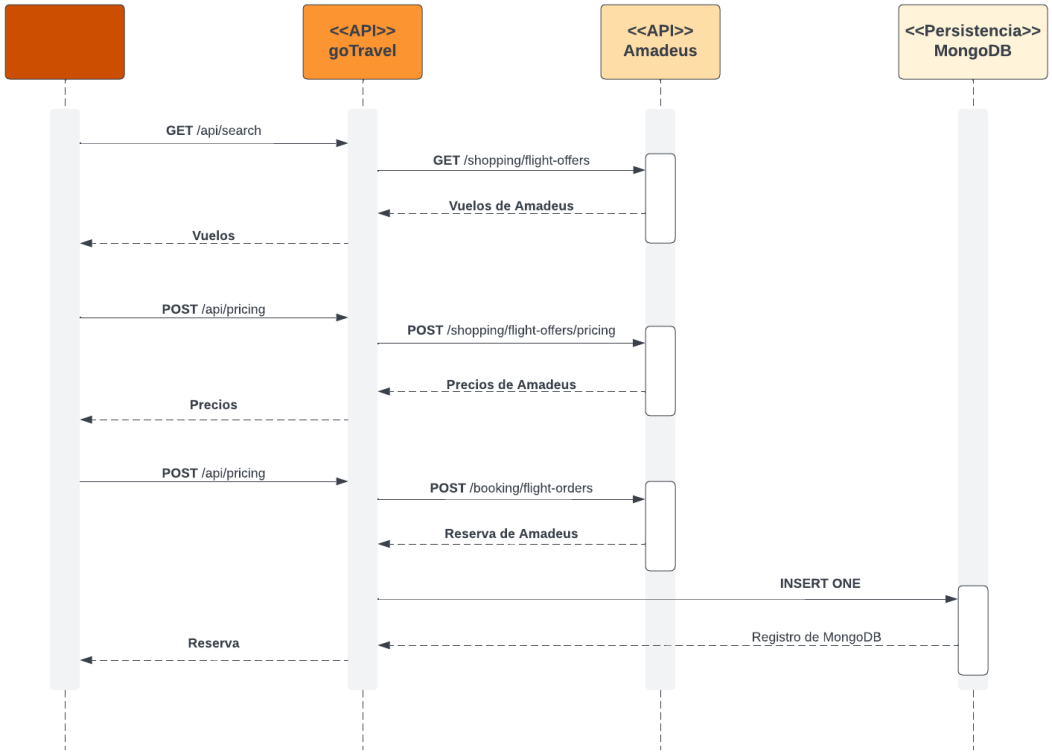
```
Seleccione un vuelo (ingrese 0 para realizar nueva búsqueda): 1
El precio total final es de: 58438.00
Pasajero 1:
Ingrese fecha de nacimiento: 1982-01-16
Ingrese nombre: Juan
Ingrese apellido: Peréz
Ingrese sexo (MALE o FEMALE): MALE
Ingrese correo: juan.perez@test.com
Ingrese teléfono: +56912345678
Reserva creada con éxito: eJzTd9cPtbSIdDYFAAryAj0
1. Realizar búsqueda.
2. Obtener reserva.
3. Salir
Ingrese una opción: 3
Gracias por usar goTravel!
```

```
Bievenido a goTravel!
1. Realizar búsqueda.
2. Obtener reserva.
3. Salir
Ingrese una opción: 2
Ingrese el ID de la reserva: eJzTd9cPtbSIdDYFAAryAj0
Resultado:
+--------+----------------+-----------------+-------+--------------+
| NÚMERO | HORA DE SALIDA | HORA DE LLEGADA | AVIÓN | PRECIO TOTAL |
+--------+----------------+-----------------+-------+--------------+
| LA103  | 11:26          | 12:21           | A320  |     58438.00 |
+--------+----------------+-----------------+-------+--------------+
Pasajeros:
+--------+----------+
| NOMBRE | APELLIDO |
+--------+----------+
| JUAN   | PERÉZ    |
+--------+----------+
1. Realizar búsqueda.
2. Obtener reserva.
3. Salir
Ingrese una opción: 3
Gracias por usar goTravel!
```

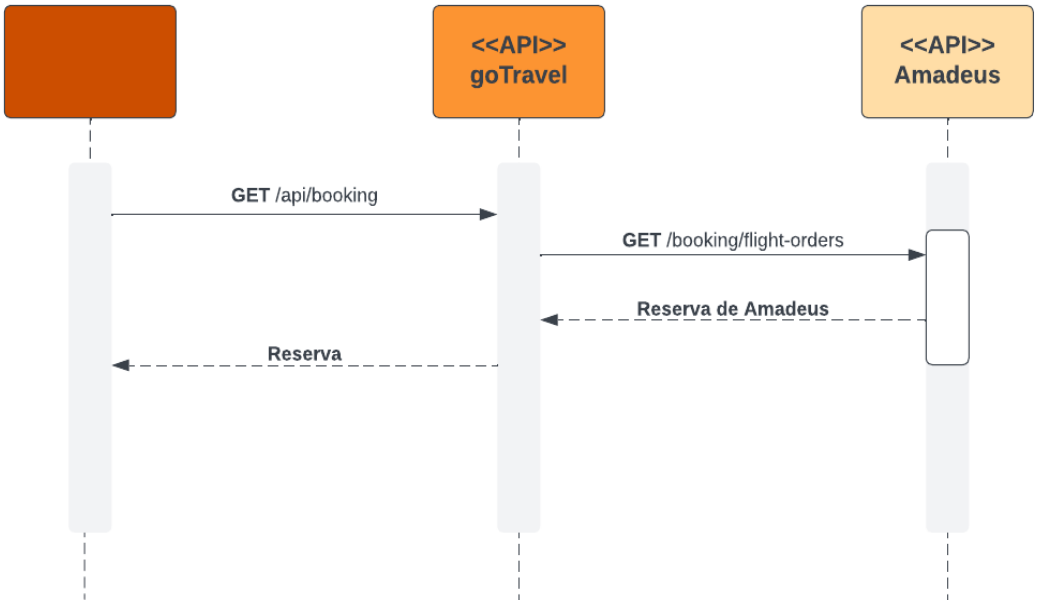
Cabe mencionar que el formato de las tablas expuestas tiene que ser desarrollado con la librería *tablewriter*.

Diagramas de secuencia

Búsqueda y creación de reserva



Obtener reserva



Programa

Dentro del directorio de la tarea, tiene que existir un archivo `main.go` que incluya la ejecución inicial del programa, cuando se ejecute, debe desplegarse un menú y en base a este menú interactuar con la API de goTravel. Otro que se requerirá, es un archivo `.env` que incluya todas las variables de entorno; como mínimo, tiene que contener las siguientes:

1. SERVER: 127.0.0.1
2. PORT: 5000
3. CONNECTION_STRING: mongodb://admin:admin@127.0.0.1:27017/gotravel
4. CLIENT_ID=xxxxxxxxxxxxxxxx
5. SECRECT_ID=xxxxxxxxxxxxxxxx

El primero y el segundo corresponden a los datos del servicio que debe desplegar con Go. El tercero, es el *string* de conexión para MongoDB. Ninguno de los valores anteriores tiene que cambiarse. El tercer archivo requerido es `server.go`, que incluye toda la lógica de la API de goTravel y que se ejecuta al levantar un servicio local en el puerto 5000, tomando los valores desde el archivo `.env`. Por último, tanto el `CLIENT_ID` como el `SECRET_ID` corresponden a las claves generadas por Amadeus y que se entregan cuando se crea una aplicación.

Observaciones

- No es necesario realizar validaciones de entrada.
- Acceda al siguiente *link* para revisar cada uno de los cURLs: https://drive.google.com/drive/folders/1nV9YqRwu3iklye869sR4i0NQVrRpah0g?usp=sharing
- Para probar las APIs, se recomienda el uso de Postman: https://www.postman.com/
- En el siguiente link se puede encontrar un tutorial para conectarse por SSH al DI: https://www.labcomp.cl/2022/04/24/tutorial-ssh/
- Aeropuertos de Chile: https://www.worlddata.info/america/chile/airports.php

REGLAS DE ENTREGA Y EVALUACIÓN

1. Toda la tarea debe ser desarrollada utilizando el lenguaje Go en su versión 1.21.X.
2. Respetar los valores para las variables de entorno.
3. Solo se realizarán dos ejecuciones para revisar la tarea:
    a. `go run main.go`
    b. `go run server.go`
4. Todo el contenido del desarrollo tiene que ser subido a la máquina virtual que se le asignará a su grupo. Allí mismo la tarea será ejecutada para su revisión.
5. La tarea tiene que ser subida en un repositorio a https://gitlab.inf.utfsm.cl y permitir acceso a la cuenta `hleyton`. La URL del repositorio debe ser registrada en el buzón que se habilitará para la entrega de la presente tarea.
6. Para dudas, escribir a hugo.leyton@usm.cl.