***Ejercicio Desarrollo C# Senior AMV Travel***

1. **Introducción**

Este proyecto tiene como objetivo desarrollar un sistema de reservas de paquetes con microservicios para la industria de viajes. Se crearán dos microservicios: uno encargado de gestionar las reservas y otro de gestionar la información de los hoteles. El proceso de reserva se realizará a través de una API REST, en ella se persistirá la información del paquete y se utilizará un intermediario de mensajes, como Kafka o RabbitMQ, para notificar al microservicio de Hoteles sobre las reservas realizadas. 

**Aclaraciones:**

* Queda a criterio propio la creación de entidades y propiedades de estas.
* Considerar la utilización de SQL Server como base de datos. Podes utilizar la misma base de datos para ambos servicios, base de datos independientes por servicio.
* Considerar utilizar Entity Framework como ORM para el servicio de Reservas. Para el servicio de hoteles puedes utilizar lo mismo, otra alternativa o incluso una base de datos no SQL.
* Aunque es necesario crear la API de reservas para probar el funcionamiento del sistema, considerar que el foco de la evaluación está en el uso del intermediario de mensajes y la comunicación entre los microservicios.
* Se recomienda utilizar una arquitectura simple para la creación de la API, ya sea en capas o similar.

1. **Instrucciones**

**1.** Crea un repositorio en Github.

**2.** Desarrolla las soluciones en C# .NET en el repositorio.

**3.** Subí los cambios a tu propio repositorio en GitHub.

**4.** Envía el enlace del repositorio. 

1. **Microservicio de Reservas**

**3.1 Modelado de Datos**

Diseña una estructura de datos para representar la información de reservas de paquetes. Incluya entidades como Cliente, Reserva, Paquete, y cualquier otra entidad relevante utilizando clases y relaciones. Considera agregar un diagrama de entidad-relación para visualizar las relaciones entre las entidades.

**3.2 Implementación del Sistema con API**

Implementa el sistema de reservas de paquetes con una API utilizando ASP.NET. Crea puntos de conexión para operaciones CRUD que consideres necesarios (Crear, Leer, Actualizar, Eliminar) en las entidades. Asegúrate de que la API sea segura y accesible a través de métodos HTTP adecuados. Proporciona una breve explicación sobre la arquitectura general del sistema y cómo se estructuran las capas.

**3.3 Documentación de la API – No excluyente**

Documente la API utilizando Swagger o algún otro framework de documentación de API en C#. Proporcione descripciones claras de los puntos de conexión, los parámetros requeridos y las respuestas esperadas.

**3.4 Pruebas Unitarias y de Integración – No excluyente**

Implemente pruebas unitarias para al menos una parte de su código. También, realice pruebas de integración para verificar la interoperabilidad de su sistema con el proveedor externo.

1. **Microservicio de Hotel**

**4.1 Integración de Microservicios**

Crear un nuevo proyecto que contenga el modelo de Hotel y realizar la integración desde el microservicio de Reservas. Diseñé e implemente un módulo que envié la información con los detalles necesarios para crear la Reserva del hotel.

**Infraestructura en Capas**

**Explicación de estructura**

El proyecto AMV Travel está organizado en varias capas para mejorar la modularidad, la separación de responsabilidades y la escalabilidad. Aquí hay una breve explicación de cada capa:

1. Controllers:

* Responsable de manejar las solicitudes HTTP y las respuestas correspondientes.
* Interactúa con los servicios para procesar la lógica de negocio.Presenta los resultados a través de las respuestas HTTP.

1. DataAccess:

* Context: Contiene la lógica de acceso a la base de datos a través de Entity Framework.
* Migrations: Almacena las migraciones de base de datos para controlar los cambios en el esquema de la base de datos.
* Seeds: Contiene datos iniciales para poblar la base de datos.

1. Models:
   * Classes: Define las clases de entidad que representan las tablas de la base de datos.
   * DTOs (Data Transfer Objects): Contiene objetos utilizados para transferir datos entre capas y servicios.
   * Enumerables: Almacena enumeraciones utilizadas en el modelo.
2. Services:
   * Kafka: Contiene la lógica relacionada con la integración y comunicación con Kafka. Proporciona métodos para enviar y recibir mensajes desde y hacia Kafka.

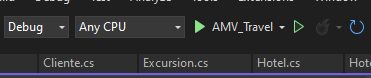
En el caso específico del proyecto AMV Travel - Hotel, hay algunas capas similares:

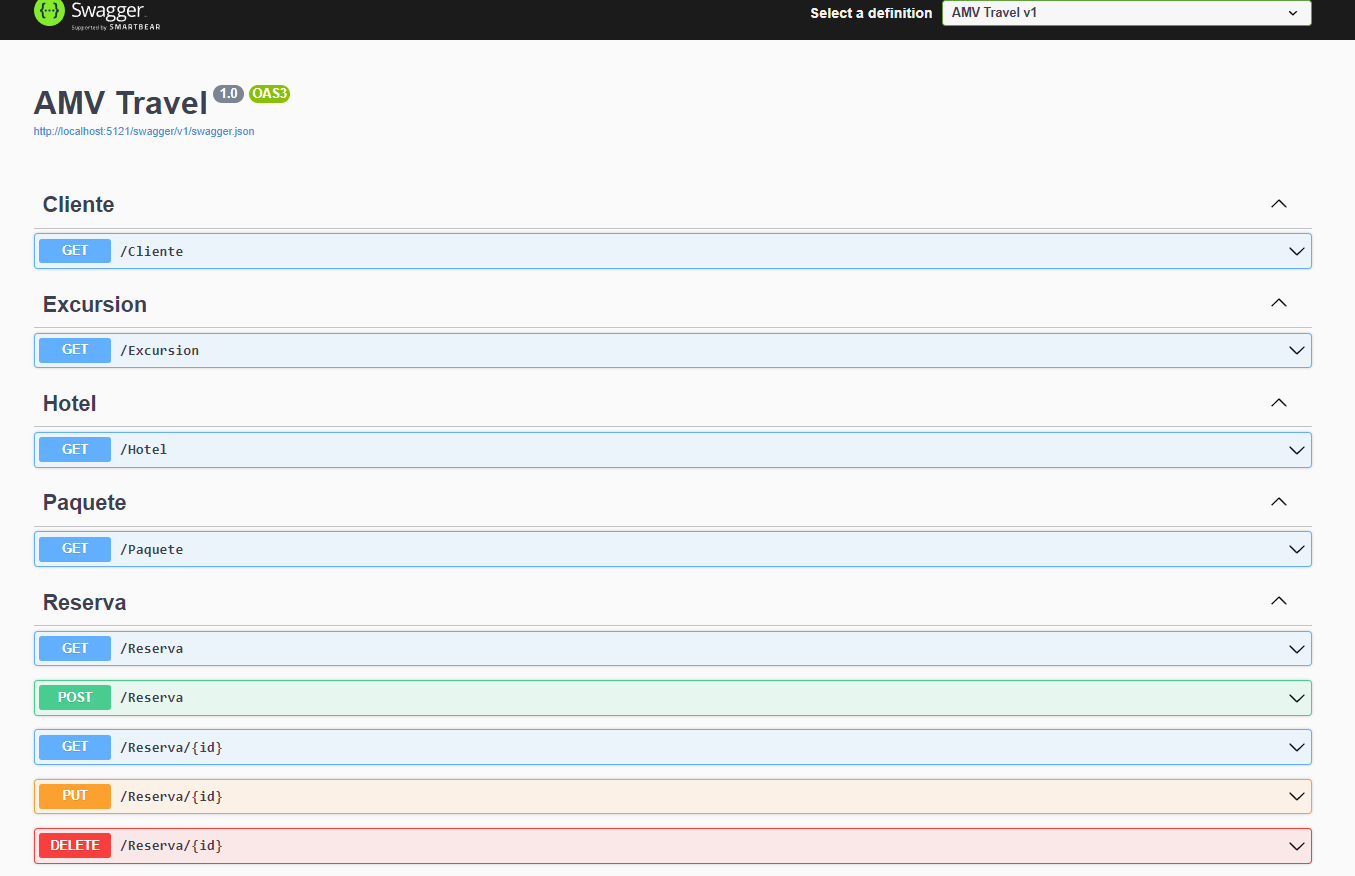
1. Config:
   * Puede contener configuraciones específicas del hotel o configuraciones adicionales necesarias para su funcionamiento.
2. DataAccess
   * Similar a la capa en el proyecto principal, contiene el contexto de la base de datos, migraciones y posiblemente datos iniciales.
3. Models:
   * Similar a la capa en el proyecto principal, define las clases y DTOs específicos del hotel.
4. Services:
   * Puede contener servicios específicos del hotel, como lógica de negocio adicional o integraciones específicas.

**Caso de uso principal**

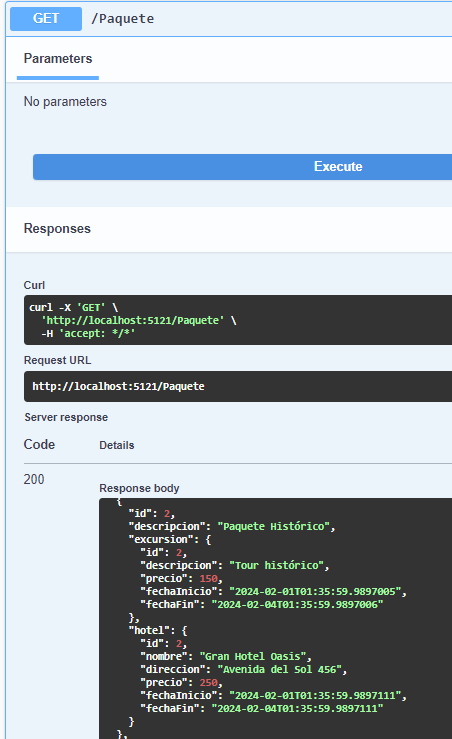
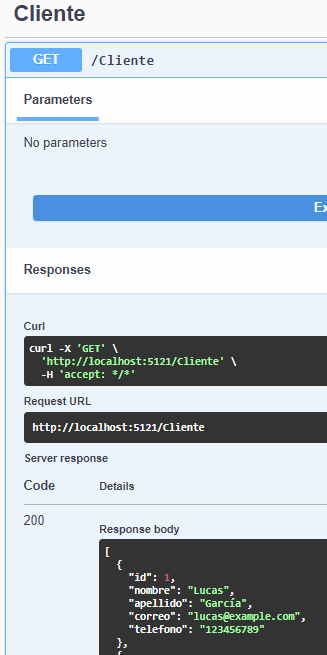
**Caso de Uso principal**

Para comenzar, ejecuta el proyecto AMV Travel. Automáticamente, se abrirá la interfaz de Swagger UI, donde encontrarás todos los endpoints disponibles.





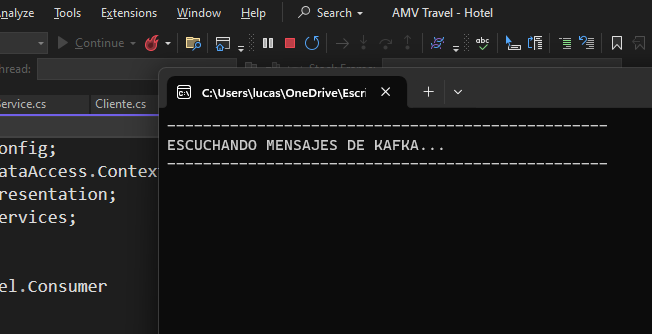
En este punto, podemos verificar los datos de un cliente y un paquete antes de proceder con la reserva.

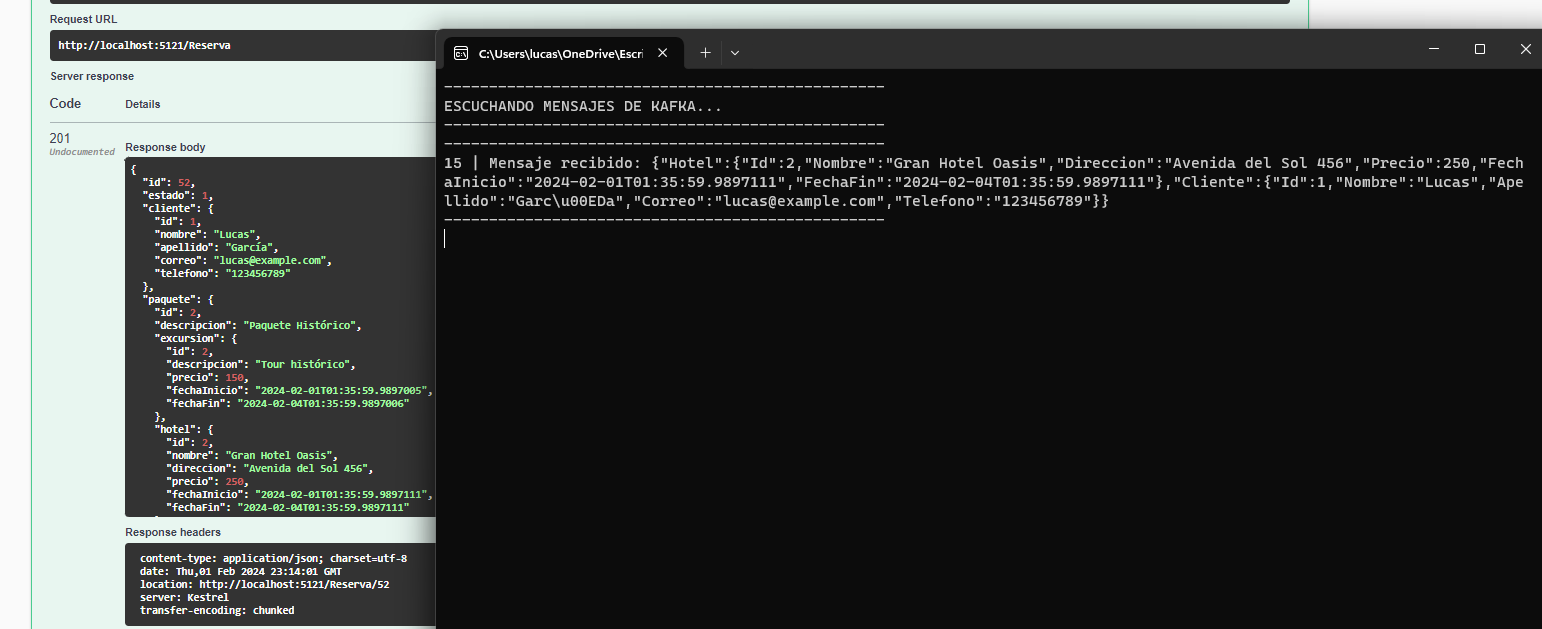
 

Rellenamos el cuerpo (body) necesario en el endpoint POST de Reserva con los datos seleccionados anteriormente.

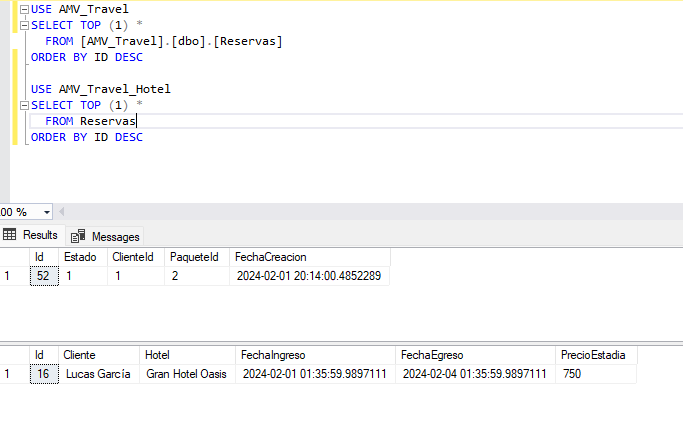


Iniciamos y ejecutamos el proyecto AMV Travel – Hotel, que simula el servidor de un hotel escuchando el tópico "amv\_travel" en Kafka.



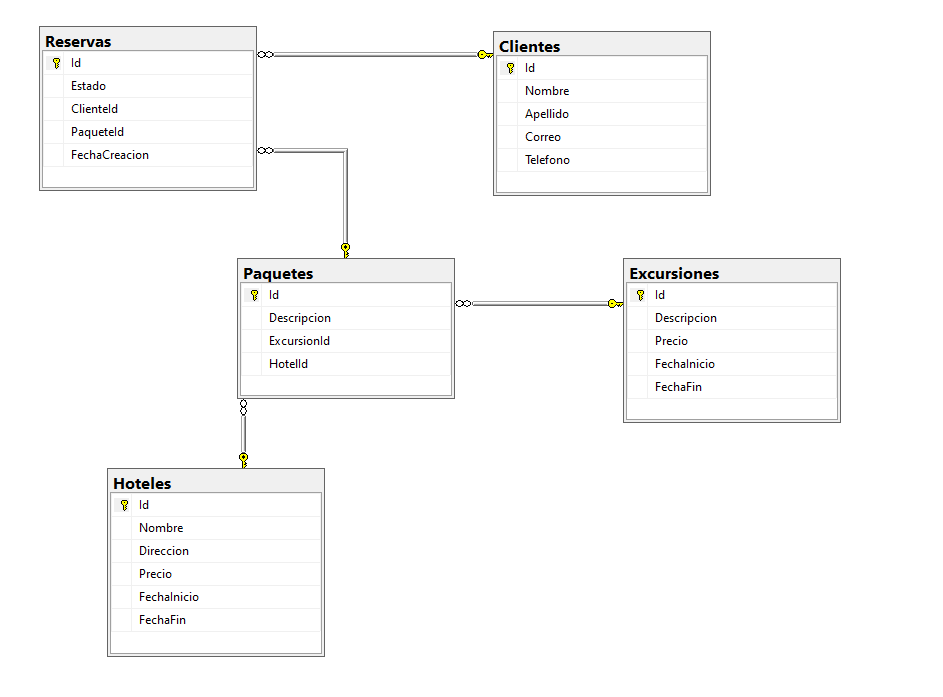
Ahora, podemos ejecutar la operación POST en Swagger. Esto nos permitirá verificar la respuesta HTTP 201, confirmando la creación de la reserva. Además, en la consola del proyecto Hotel, observaremos el mensaje consumido a través de KAFKA.  
  


Adicionalmente, la información de la reserva se guarda en ambas bases de datos.



**DER**

**Caso de Uso principal**

****