PROPÓSITO

El propósito de esta documentación es hacer referencia a todo lo relacionado con el ejercicio llamado JavaTechAssignment. En esta documentación se encontrarán algunos análisis del entorno de trabajo, así como la generación previa de la aplicación que se realizó.

También se detallarán los casos en los que se podrá utilizar esta aplicación, las herramientas que se utilizaron y algunos conceptos que se tienen que tener en cuenta.

ENUNCIADO

Usted ha sido contratado como desarrollador backend para una importante empresa de turismo, actualmente el registro de las reservas se realiza a través de correo electrónico o llamadas telefónicas con asesores, lo que ha causado molestias a algunos clientes debido a que algunas solicitudes requieren de mucho tiempo para su confirmación.

El área de tecnología requiere implementar un sistema que permita el registro de las reservas usando un message bróker (Kafka, Rabbit MQ, etc) lo que reducirá el tiempo de espera actual. Se debe tener en cuenta un buen manejo de los posibles errores ya sea por un mensaje con información inválida o posibles problemas en la red.

PASOS INICIALES

Se recomienda utilizar Visual Studio Code, IDE utilizado para crear el aplicativo. Lo primero que se realiza antes de ejecutar el programa es instalar las siguientes herramientas:

- Node JS
- PostgreSQL
- RabbitMQ
- PostMan

Las extensiones utilizadas dentro de Visual Studio Code fueron las siguientes:

- Extension Pack for Java
- Spring Boot Extension Pack

CÓDIGO

En este apartado se explicarán los principales archivos que tiene el programa, así como su función.

- Pom.xml:

Aquí encontraremos todas las dependencias que se utilizaron en el desarrollo del aplicativo, entre estas están:

  - SpringBoot JPA
  - SpringBoot Web
  - SpringBoot AMQP
  - SpringBoot WebTools
  - PostgreSQL
  - Lombok

- Application.properties:

Aquí encontramos algunas variables generales del programa, como el usuario y contraseña de Postgres y RabbitMQ. Los elementos que se tienen que cambiar son los anteriores mencionados dependiendo de la configuración que se hizo a la hora de instalar estas dos herramientas.

- ReservaController:

Aquí encontramos el controlador del API, donde están las funciones de GET y POST. De aquí nos redirigimos al servicio para hacer las respectivas funciones.

Encontramos un GET("/consultar-reserva") que nos muestra todas las reservas hechas, un GET("/consultar-reserva/{id}") que nos muestra la reserva hecha con un id, y por último tenemos un POST("/registrar-reserva") que nos permite crear una reserva y guardarla en la base de datos.

- Reserva:

Aquí tenemos nuestro Productor, es decir, el archivo donde se crean las reservas. Nos encontramos con las variables que se tienen en cuenta a la hora de hacer el POST, como *fechaIngreso, fechaSalida, totalDias, numeroPersonas, titularReserva, numeroHabitaciones, numeroMenores.*

- ReservaRepository:

Aquí hacemos uso de la librería JPA para hacer más simple la subida de la información a la base de datos, que en nuestro caso es PostgreSQL. Simplemente se extiende la clase hacia la librería para acoger todas las funciones.

- ReservaService:

Aquí encontramos las funciones que se realizan al llamar a un POST o un GET. Se utiliza el ReservaRepository para cumplir las funciones de subir y recibir la información de la base de datos, y también se utiliza RabbitMQ para hacer una cola de mensajes y que si en algún momento se realizan varios POST a la vez no se sobrecargue el aplicativo.

EJEMPLOS

Al realizar los procesos anteriores, ejecutamos el programa y hacemos uso de PostMan para validar que los endpoints se estén utilizando correctamente:

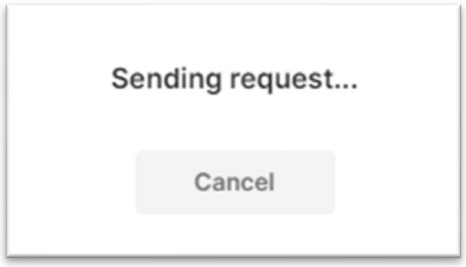
- POST("/registrar-reserva")

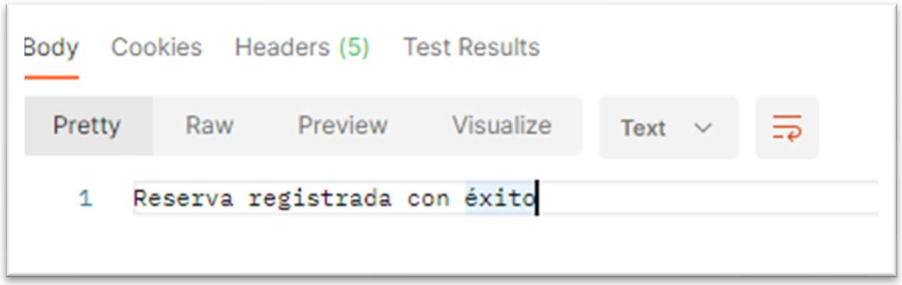
Para este caso, utilizaremos algunos JSON que nos servirán para subir la información a la base de datos. La url utilizada es:

localhost:8080/api/registrar-reserva

```
{
    "fechaIngreso":"2021-12-23",
    "fechaSalida":"2021-12-25",
    "totalDias":3,
    "numeroPersonas":5,
    "titularReserva":"Daniel Rubiano",
    "numeroHabitaciones":2,
    "numeroMenores":2
}
```

Este código se pasa como un raw dentro del Body, dando el siguiente resultado:





En la siguiente petición nos daremos cuenta que si se registró correctamente.

- GET("/consultar-reserva")

Para este caso, sólo se tendrá que poner la siguiente url para recibir todas las reservas que se han realizado:

localhost:8080/api/consultar-reserva/

Obteniendo lo siguiente:


```
[
    {
        "id": "d344f110-1903-41a3-9d6f-0d5bb31567a1",
        "fechaIngreso": "2021-12-23",
        "fechaSalida": "2021-12-25",
        "totalDias": 3,
        "numeroPersonas": 5,
        "titularReserva": "Daniel Rubiano",
        "numeroHabitaciones": 2,
        "numeroMenores": 2
    }
]
```


Vamos a agregar otra reserva más utilizando la petición POST explicada anteriormente, con los siguientes parámetros

```
{
    "fechaIngreso":"2022-02-17",
    "fechaSalida":"2022-02-24",
    "totalDias":8,
    "numeroPersonas":12,
    "titularReserva":"Alan Turing",
    "numeroHabitaciones":4,
    "numeroMenores":3
}
```


Ahora, al utilizar el GET nos muestra lo siguiente:


```
[
    {
        "id": "d344f110-1903-41a3-9d6f-0d5bb31567a1",
        "fechaIngreso": "2021-12-23",
        "fechaSalida": "2021-12-25",
        "totalDias": 3,
        "numeroPersonas": 5,
        "titularReserva": "Daniel Rubiano",
        "numeroHabitaciones": 2,
        "numeroMenores": 2
    },
    {
        "id": "dde26476-7bf5-4d42-9fe7-4709402e6715",
        "fechaIngreso": "2022-02-17",
        "fechaSalida": "2022-02-24",
        "totalDias": 8,
        "numeroPersonas": 12,
        "titularReserva": "Alan Turing",
        "numeroHabitaciones": 4,
        "numeroMenores": 3
    }
]
```


- GET("/consultar-reserva/{id}")

Teniendo algún ID de reservas que se hayan realizado, se pueden mostrar individualmente agregándole éste dentro del URL, por ejemplo:

localhost:8080/api/consultar-reserva/ dde26476-7bf5-4d42-9fe7-4709402e6715

En este caso sólo nos va a mostrar la reserva que se le asocia a este id, así:

```
{
    "id": "dde26476-7bf5-4d42-9fe7-4709402e6715",
    "fechaIngreso": "2022-02-17",
    "fechaSalida": "2022-02-24",
    "totalDias": 8,
    "numeroPersonas": 12,
    "titularReserva": "Alan Turing",
    "numeroHabitaciones": 4,
    "numeroMenores": 3
}
```