Wipay

Tareas Prácticas

Todo tiene que estar documentado aprovechando el uso de los conventional commits

1. Creación de dos microservicios

Vamos a crear 2 microservicios, client y merchant, la idea general es poder hacer operaciones básicas con mercados y clientes respondiendo a preguntas como ¿cuáles son los mercados de un cliente X? o ¿a quién pertenece el mercado con nombre X?

- Versiones
    - Java: 8

      Springboot: 2.2.5.RELEASE
- Dependencias obligatorias
    - lombok
    - mapstruct
    - dynamodb (tendremos que montar una dynamoDB en local y conectarla con nuestros microservicios empleando únicamente 1 tabla llamada MainTable, podéis aprovechar Docker para esto)
    - swagger

2. Preparación y primeros endpoints

Por ahora vamos a utilizar la arquitectura MVC (Modelo Vista Controlador), aunque la vamos a adaptar un poco de tal forma que:

- Carpeta controller para nuestros controladores
- Carpeta service para nuestras interfaces e implementaciones
- Carpeta model para modelos generales, de entrada y de salida
- Carpeta repository para los repositorios
- Carpeta mappers para todo lo relevante a mapstruct

Detalles

- Todas las entidades van a extender de la clase principal “MainTable”

- Todas las entidades, modelos, controladores, servicios y repositorios van a utilizar de lombok para la creación de constructores, getters, setters e inyección de dependencias (sin usar @Autowire)
- Los controladores siempre que se pueda recibirán un modelo de entrada y devolverán un modelo de salida
- Los servicios siempre van a devolver y recibir el modelo genérico necesitado
- Todos los mapeos se deberán de hacer con mappers utilizando mapstruct

Endpoints

- EP para crear un cliente y merchant
- EP para visualizar un cliente y merchant por ID, nombre y email en caso de cliente
    - findById
        - Aparte del modelo de entrada un parámetro que determina si le devolveremos un modelo de salida con todos los datos o solo con la ID, es decir: si mandamos un parámetro que indique “simpleOutput”, el EP solo devolverá la ID, sin embargo, si no le mandamos nada, devolverá el modelo completo.
    - findByName
        - Debe devolver las coincidencias, es decir, si tenemos un cliente cuyo nombre es “Wipay” y al endpoint le pasamos “pay”, deberá de devolver todas las coincidencias que en el nombre tengan un “pay” independientemente de las mayúsculas o minúsculas
    - findByEmail
        - Importante controlar que el email está bien formado en el modelo de entrada mediante la etiqueta del @Pattern
- EP para poder saber los merchants de un cliente como también a que cliente pertenece un merchant
- EP para modificar cliente y merchant

Modelos

- MainTable tendrá los siguientes atributos: PK, SK, id, status, gIndex2Pk, createdDate
- Client tendrá los siguientes atributos: name, surname, cifNifNie, phone, email
- Merchant tendrá los siguientes atributos: name, address, merchantType
    - merchantType es un Enum con valores como: MERCHANT_TYPE_PERSONAL_SERVICES, MERCHANT_TYPE_FINANCIAL_SERVICES

3. Feign

Vamos a conectar ambos microservicios mediante la dependencia Feign que nos va a permitir poder llamar desde un microservicio a otro para utilizar alguna llamada, como por ejemplo:

- Creación de un EP en el microservicio de Client que pasándole una ID de un merchant, nos va a indicar si este existe llamando al servicio de “findById” que estaría definido en el MS de Merchant

4. Interceptor (usando HandlerInterceptor)

Tendréis que montar un interceptor usando HandlerInterceptor de tal forma que podáis como el propio nombre indica, interceptar las peticiones antes de que lleguen al controlador para obtener algo de interés de la request. Tendréis que:

- Crear el interceptor
- Crear un jwt con un nombre y una edad, simulando la persona que está mandando la petición
- Mandamos el jwt por parámetros de la request y captamos ese parámetro en el interceptor para leerlo, si la edad es menor de 18, no dejaremos que la petición llegue al controlador saltando una excepción, en caso contrario, si llegará al controlador