## Microservicios con Spring y Maven

En este espacio buscamos organizar, mostrar y «registrar» lo más similar y posible a un paso a paso sobre cómo fuimos planeando y llevando a cabo nuestro proyecto de finanzas SPEN2.

Iniciamos con la planeación inicial de los microservicios que llevará y formarán lo que será finalmente la plataforma. Para finalmente poder llegar a los siguientes requerimientos del sistema, se realizó una lluvia de ideas, donde se destacó todo lo relacionado con el manejo y administración de finanzas, primeramente se realiza un registro de usuario, luego pasa a utilizar la organización de ingresos y egresos por categorías, un apartado para revisión de transacciones, entre otras cosas.

### 

### A. Implementación de Microservicios:

Como ya se mencionó anteriormente, los microservicios manejados por el sistema son la existencia de «categorías», que permite seleccionar y presentar las acciones con sus respectivas disponibilidades existentes en nuestro sistema, además de, permitir ordenar cada objeto dentro de la aplicación; el poder efectuar una «transacción», lugar que como su nombre lo dice, es donde podemos transaccionar (enviar o recibir) dinero, derivando este mismo a cualquier meta de ahorro o persona estimada que uno quiera y requiera; el historial de «movimientos», en este lugar somos capaces de ver cada uno de los movimientos que ha tenido el dinero de los usuarios dentro del sistema, pudiendo conocer la fecha, estimación y monto utilizado en el movimiento monetario requerido; y por último, la opción de crear y organizar «metas de ahorro», que como su nombre lo dice, permite crear y personalizar tanto como se pueda cualquier meta de ahorro que uno quiera, donde uno puede delimitar la cantidad de dinero que uno espera acumular.

La comunicación entre ellas utilizando APIs RESTfull fue posible gracias a la existencia de métodos como Get, dónde obtenemos datos almacenados en la base de datos; Post, donde enviamos datos para que sean almacenados dentro de la base de datos; Put, que es para modificar «todo» el contenido relacionado a un objeto, a diferencia de Patch que modifica solo una pequeña parte sobre el objeto elegido; delete se encarga de eliminar completamente los datos de un objeto almacenado en la base de datos.

### **B. Base de Datos:**

Cada uno de los microservicios creados debe y tiene su propio espacio de funcionamiento en la base de datos, lo que garantiza totalmente su independencia de funcionamiento al momento de inicializar el programa, lo que permite que la plataforma no caiga por completo en caso de que alguna de sus funciones presente algún error.

La seguridad de los datos es posible y garantizada por la forma en la que utilizamos sentencias como JPA, que incluye opciones como @entity, @id, entre otras para garantizar que los datos estén completos y correctamente organizados e ingresados antes de ser almacenados en la propia base de datos.