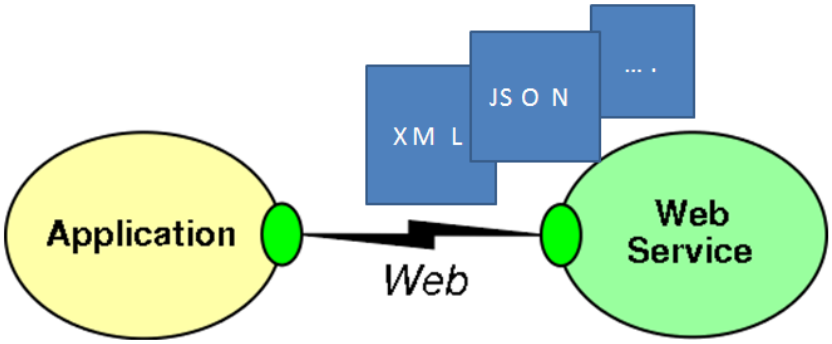
ARQUITECTURA ORIENTADA A SERVICIOS

PROFESORA: MARCELA CASTRO



David Feijoo y Mouctar Diallo

19 de noviembre de 2018

1) **Diseñar y desarrollar un servicio web ReST que se utilice para gestionar el repositorio de datos relativa a la a.**

Grupo 1 (Mouctar/David): Gestión de ingresos y egresos a una cuenta bancaria.

2) **Definir el modelo de operaciones, indicando cada una de las operaciones, sus parámetros de entrada y de salida y una breve descripción. La definición no debería limitarse a la dificultad de la implementación, sino que se debe indicar a su criterio que funciones API se requerirán para la gestión del repositorio de datos.**

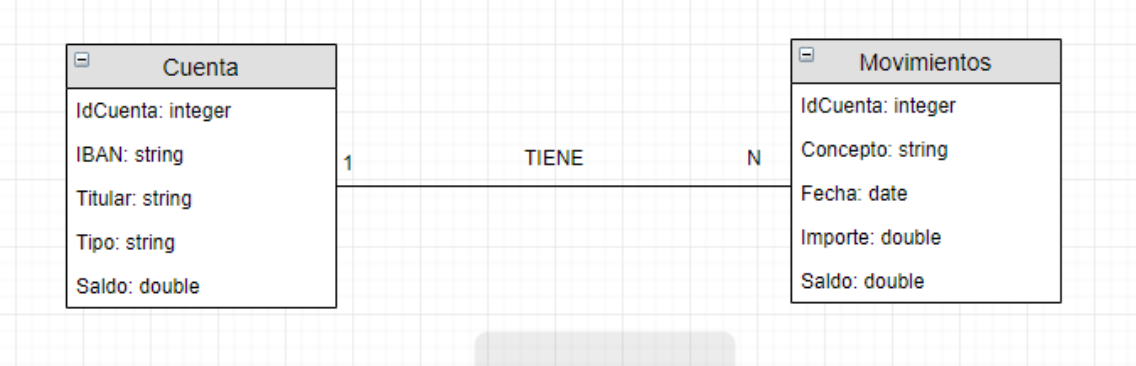
Nuestro modelo de operaciones es el siguiente:

Descripción	Operación/URL	Parámetros de entrada	Salida OK
Alta de una cuenta	POST /cuenta	IBAN, Titular, Tipo, Saldo	IdCuenta
Modificar un atributo: solo titular / saldo	PATCH /cuenta	Titular / Saldo	IdCuenta + IBAN, Titular, Tipo, Saldo
Listar movimientos De una cuenta	GET /cuenta/{id cuenta}/movimientos	NONE	IdCuenta, Concepto, Fecha, Importe, Saldo actual
Añadir un registro de ingreso/egreso	POST / cuenta/{idCuenta}/movimientos	Concepto, fecha, importe, saldo actual	Id_movimiento, + concepto, fecha, importe, saldo actual
Listar cuentas de un titular	GET / cuenta/{Titular}	NONE	IdCuenta, IBAN, Titular, Tipo, Saldo

En esta tabla de operaciones tenemos operaciones tanto de cuenta como de movimiento. En la práctica hemos implementado la clase cuenta con sus atributos y podemos realizar operaciones como dar de alta una cuenta, con la comanda POST, modificar atributos de las cuentas y listar cuentas a partir del nombre de su titular.

3) Definir el modelo de entidades para implementar el servicio. Indicar las entidades y las relaciones entre ellas indicando si son relaciones 1 a 1 (OnetoOne), 1 a N (OneToMany), N a 1 (ManyToOne) o ManyToMany (N a N).

Nuestro modelo de entidades es el siguiente. Como vemos tenemos dos entidades (Cuenta y Movimientos) y tiene una relación de 1 a N (OnetoMany), es decir una cuenta tiene varios movimientos y un movimiento pertenece a una cuenta.



4) Implementar las operaciones utilizando Spring con JPA y base de datos Mysql. Si las operaciones definidas son muchas y/o complejas, se acordará con el profesor cuales implementar.

Nuestro proyecto tiene una clase Cuenta y una clase interface CuentaRepository. La clase cuenta tiene los atributos Idcuenta, iban, titular, tipo y saldo. En CuentaRepository tenemos las operaciones que podemos realizar con el WebRest, en nuestro caso tenemos la de listar cuentas a partir de un titular que pasamos por parámetro.

5) Probar las operaciones implementadas. Desarrollar y entregar un script basado en "Curl" que permita dar de alta varios registros, modificar y borrar datos.

Hemos realizado un script donde están todas las operaciones que hemos implementado. Hemos añadido nuevas cuentas, hemos modificado atributos de las cuentas y hemos eliminado cuentas. Por último, hemos realizado una búsqueda de las cuentas a través de un titular. El script se encuentra dentro del proyecto.

Conclusión: Con esta práctica hemos entendido mejor todos los elementos que se usan para realizar un WebRest. También hemos entendido bien todas las operaciones que nos permite realizar. Es una buena práctica para probar el funcionamiento de un WebRest.