5) Dada una tabla CLIENTES con los campos CLIENTE y SALDO, se solicita implementar un método que mediante dos updates reste un importe del campo saldo del cliente1 y lo sume al saldo del cliente en una sola transacción, de forma que en caso de excepción el saldo de los dos clientes vuelva a su valor anterior.

a) Las tablas con utilizadas son las siguientes cuenta y persona (en vez de cliente)

CREATE TABLE `accounts` (

`idaccount` int NOT NULL AUTO\_INCREMENT,

`idPersona` int NOT NULL,

`monto` decimal(10,2) DEFAULT '0.00',

`nro\_cta` varchar(10) NOT NULL,

PRIMARY KEY (`idaccount`),

UNIQUE KEY `nro\_cta\_UNIQUE` (`nro\_cta`)

) ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `persona` (

`id\_persona` int NOT NULL AUTO\_INCREMENT,

`nombre` varchar(45) DEFAULT NULL,

`apellido` varchar(45) DEFAULT NULL,

`email` varchar(45) DEFAULT NULL,

`telefono` varchar(45) DEFAULT NULL,

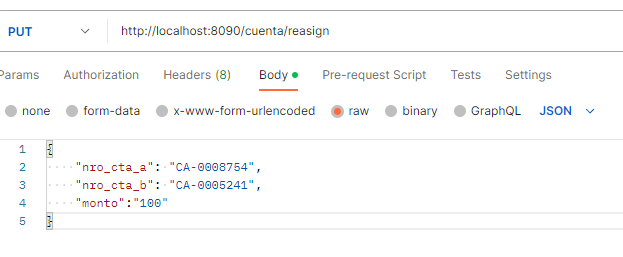
PRIMARY KEY (`id\_persona`)

) ENGINE=InnoDB AUTO\_INCREMENT=10 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

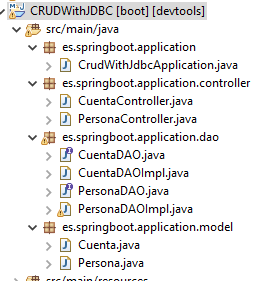
insert into test.accounts (idPersona,monto,nro\_cta) values ('1',2563.25,'CA-0008754');

insert into test.accounts (idPersona,monto,nro\_cta) values ('2',5419.73,'CA-0005241');

b) Luego para la prueba se utilice el siguiente put



c) Se utiliza la siguiente estructura de API REST



d) Desde el CuentaController se utiliza el siguiente método

@PutMapping("/reasign")

**public** String reasignaMontoOtraCuenta(@RequestBody String reasignaVal) {

**int** registros=0;

Object obj = JSONValue.*parse*(reasignaVal);

JSONObject jsonObject = (JSONObject) obj;

Cuenta ctaA = cuentaDAO.getByNroCta((String) jsonObject.get("nro\_cta\_a"));

Cuenta ctaB = cuentaDAO.getByNroCta((String) jsonObject.get("nro\_cta\_b"));

String strmonto = (String) jsonObject.get("monto");

**if**(ctaA == **null** || ctaB == **null** || strmonto.isEmpty()) {

**return** "Revise los parámetros ingresados...";

}

**else** {

Double monto = Double.*parseDouble*(strmonto);

**try** {

registros = cuentaDAO.reasignaMontoOtraCuenta(ctaA, ctaB, monto);

}

**catch**(Exception e) {

e.printStackTrace();

}

}

**return** "Se han modificado: " + registros + " registros.";

}

e) Desde CuentaDAO se utiliza el siguiente método con @Transactional para poder realizar un rollback automático en caso de error

@Transactional

@Override

**public** **int** reasignaMontoOtraCuenta(Cuenta ctaA, Cuenta ctaB, Double monto) {

**int** ret = 0;

ret = jdbcTemplate.update("update test.accounts set monto = ? where nro\_cta = ?",Double.*toString*(ctaA.getMonto()-monto),ctaA.getNro\_cta());

//Test para rollback

//ctaB = null;

ret += jdbcTemplate.update("update test.accounts set monto = ? where nro\_cta = ?",Double.*toString*(ctaB.getMonto()+monto),ctaB.getNro\_cta());

**return** ret;

}

**Se adjunta CRUD entero para poder verlo con mayor detalle**