**EXAMEN DE RECUPERACIÓN DESARROLLO APLICACIONES WEB ENTORNO SERVIDOR**

**Junio 2023**

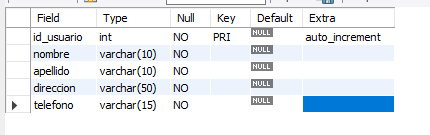
**Tiempo: 4 horas**

**IMPORTANTE: LEER NOTAS AL FINAL DEL ENUNCIADO**

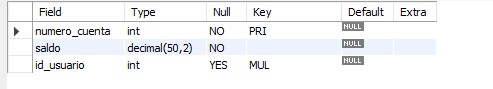
**Ejercicio: Aplicación REST Banco**

Tenemos una BBDD que representa a un banco con dos tablas relacionadas: clientes y cuentas. Podéis descargar la BBDD de [aquí](https://calasanzescolapiossalamanca-my.sharepoint.com/:u:/g/personal/kikesainz_calasanzescolapiossalamancao365_educamos_com/ESRU84erhRFJize2Bky3QYYBLEAAFfV16Jy1Zh2frEwZBA?e=9tgpb6).

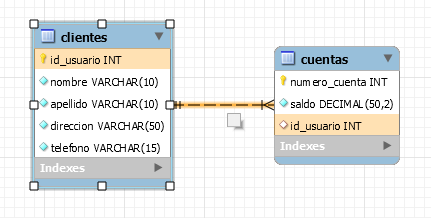
**Clientes:**



**Cuentas:**



\*id\_usuario es la FK



Se pide crear una aplicación que se llame banco y que permita las siguientes operaciones:

* **CRUD completo de la tabla clientes usando una arquitectura REST:**
  + **GET** [http://localhost:8080/banco/clientes](http://localhost:8080/colegio/clientes) nos mostrará todos los clientes del banco con su cuenta y saldo asociados. Es decir, el JSON tiene que tener este formato **(1 punto)**:



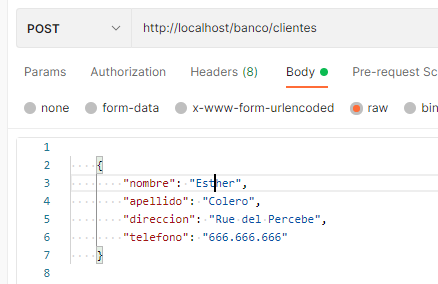
* + Además, el GET permitirá filtrar por nombre y apellidos usando los query params nombre y apellido **(1 punto),** es decir, la llamada:



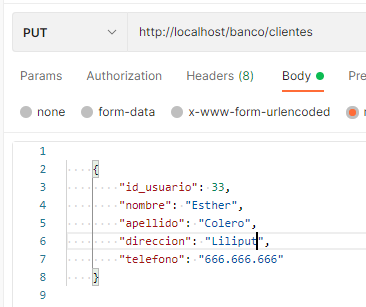
Daría el siguiente resultado:



* + **POST:** Hará un insert en la tabla **clientes (0,5 puntos):**



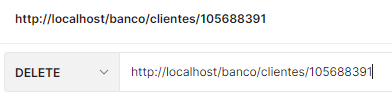
* + **PUT:** Hará un update de un registro ya existente **(0,25 puntos):**



* + **DELETE:** Hará un delete del registro de la BBDD **(0,25 puntos)**. El formato será:

http://localhost:8080/banco/clientes /{id}

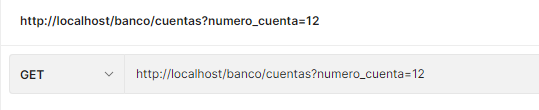
Ejemplo:



* **READ de la tabla cuentas usando una arquitectura REST:**
  + **GET** [http://localhost:8080/banco/cuentas](http://localhost:8080/colegio/cuentas) nos mostrará todas las cuentas del banco con su nombre de cliente asociado **(1 punto):**



* + Además, el GET permitirá filtra por nombre y apellidos usando query params **(1 punto),** es decir, la llamada:



Devolverá:



La implementación de los CRUDs se podrá hacer usando JDBC, Hibernate o Spring Data JPA. **Si el DAO se implementa de dos formas distintas sumará 3 puntos.**

* **Cálculo de la comisión. (2 puntos)**

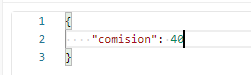
**Se tendrá que programar la siguiente API:**

[http://localhost/banco/comision?id\_cliente=33](http://localhost/banco/comision&id_cliente=33)

Que calculará la comisión en base a los siguientes criterios:

* Si el cliente tiene una sola cuenta y un saldo menor de 1000€: 40€.
* Si el cliente tiene una sola cuenta y un saldo mayor de 1000€: 20€
* Si el cliente tiene más de una cuenta y un saldo mayor de 1000€: 10€

La salida de la API tiene que ser de la siguiente forma:



La lógica del cálculo de la comisión tiene que estar en una capa de servicio y las consultas a la BBDD en otra de acceso a datos. (si no se hace así restará 1 punto).

**NOTAS IMPORTANTES:**

* El proyecto se debe de llamar Banco. No respetar esta regla restará 1 punto.
* El proyecto se debe entregar usando el repositorio que tenemos para las prácticas y subirlo a github. No hacerlo así restará 2 puntos.
* El context-path de la aplicación será /banco. No usar este context-path restará 1 punto.
* La entrega constará, además del proyecto, del export de la colección de la API POSTMAN. No hacerlo así restará 1 punto.
* Se puede usar internet, pero no está permitida la comunicación con ninguna otra persona o grupo. No se puede usar tampoco ningún tipo de IA.
* Debe estar activo el Veyon. Si no está activo el Veyon no se podrá usar internet.