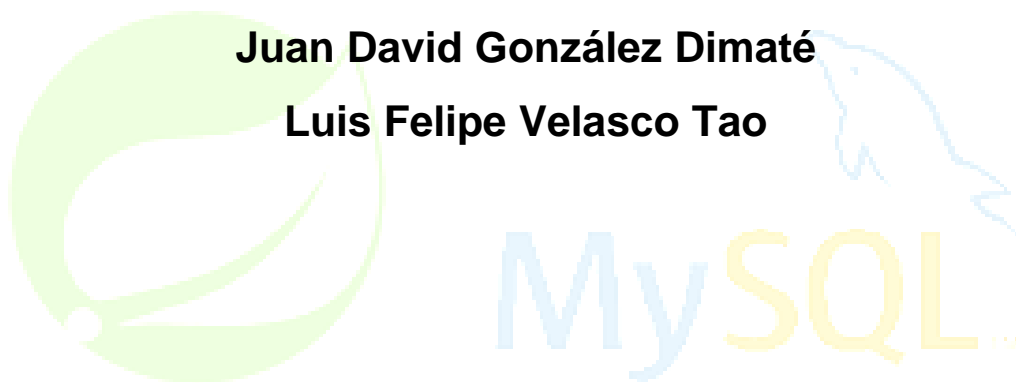


# **Desarrollo de APIs para la manipulación de una base de datos del Banco**

**Juan David González Dimaté**

**Luis Felipe Velasco Tao**



**Universidad de San Buenaventura**


**Programa de Ing. De Sistemas**

**Sistemas Operativos**

**Facultad de Ingeniería**


**Bogotá D.C**

**2021-1**


 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------

## Tabla de Contenido


Desarrollo de APIS para un Banco	8
Descripción de la actividad	8
Objetivos	8
Objetivo General	8
Objetivos específicos	8
Software utilizado	8
Instalación de software	10
Instalación de MySQL 5.7	10
Instalación de MySQL Workbench 8.0	17
Instalación de Java 1.8	22
Instalación de Eclipse Enterprise Edition	23
Instalación de Spring Boot 4	24
Instalación de Postman	25
Cuenta de Github	26
Instalación de GitHub Desktop	27
Desarrollo	28
Distribución de la base de datos	28
Análisis	28
Objetivos	28
Objetivo General	28
Objetivos específicos	28
Diseño conceptual	29
Esquema conceptual global	30
Diseño de las vistas	31
Usuarios	31

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------


Información de acceso	32
Definición de esquemas externos	32
Diseño de la distribución	32
Cuentas clientes	33
Transacción	34
Esquemas conceptuales locales	34
BD Central	34
BD Bogotá	35
BD Medellín	35
Diseño físico	36
BD central	36
BD Bogotá	37
BD Medellín	37
Esquema físico	38
Creación base de datos	40
Banco Bogotá DDL	40
Banco Medellín DDL	41
Datacenter DDL	43
Creación de repositorio	45
Clonación del repositorio	49
Manipulación del repositorio	51
Creación APIs	54
Creación del proyecto	54
API Banco Bogotá	57
Application.properties	57
Model	57

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------


Ciudad _____	57
Cuenta _____	59
Tipo_Transaccion _____	62
Transacción _____	64
Repository _____	67
ICiudad _____	67
ICuenta _____	67
ITipoTransaccion _____	68
ITransaccion _____	68
Rest _____	68
CiudadController _____	68
CuentaController _____	70
TipoTransaccionController _____	72
TransaccionController _____	74
API Banco Medellín _____	76
Application.properties _____	76
Model _____	76
Ciudad _____	76
Cuenta _____	76
Tipo_Transaccion _____	76
Transacción _____	76
Repository _____	76
ICiudad _____	76
ICuenta _____	76
ITipoTransaccion _____	76
ITransaccion _____	76
Rest _____	76
CiudadController _____	76
CuentaController _____	78

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

TipoTransaccionController	80
TransaccionController	82
API Banco Datacenter	84
Aplication.properties	84
Model	84
Ciudad	84
Cuenta	84
Tipo_Transaccion	84
Auditoria	84
Repository	84
ICiudad	84
ICuenta	84
ITipoTransaccion	84
IAuditoria	84
Rest	84
CiudadController	85
CuentaController	86
TipoTransaccionController	88
AuditoriaController	90
<b>Pruebas</b>	<b>92</b>
Requests Ciudad	92
Create	93
Find All	93
Find by ID	93
Count	94
Update	94
Delete	94
Tabla de métodos	95


 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------

Requests TipoTransaccion_____	95
Create _____	96
Find All _____	96
Find by ID _____	96
Count _____	97
Update _____	97
Delete_____	97
Tabla de métodos_____	98
Requests Cuenta _____	98
Create _____	99
Find All _____	99
Find by ID _____	100
Count _____	100
Update _____	100
Delete_____	101
Tabla de métodos_____	101
Requests Transaccion _____	103
Create _____	103
Find All _____	103
Find by ID _____	104
Count _____	105
Update _____	105
Delete_____	105
Tabla de métodos_____	106
Requests Auditoria_____	107
Create _____	107

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------

Find All _____	108
Find by ID _____	109
Count _____	110
Update _____	110
Delete _____	111
Tabla de métodos _____	111
Conclusiones _____	113
Anexos _____	113
Repositorios de APIs desarrolladas _____	113



 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
--	---	--	----------

## Desarrollo de APIS para un Banco

En este documento se describirá todo el proceso para el despliegue de una base de datos, basándonos en el ejercicio previamente realizado para la distribución de la base de datos, enfocándonos en la creación de APIs encargadas de la manipulación de los datos contenidos en cada distribución.

### Descripción de la actividad

Se requiere la creación de una base de datos para un banco, el cual requiere la gestión de las cuentas de sus clientes y las transacciones que estos realizan. El banco cuenta con dos sedes, una en Bogotá y otra en Medellín, y nos da a conocer que tienen una central en donde se realizan las auditorías a las cuentas. Para poder realizar la correcta manipulación de los datos, se requiere realizar la distribución de las bases de datos, e implementar algún componente que se encargue de la gestión de las distribuciones de las bases de datos, tal cual como si fuera una base de datos centralizada.

### Objetivos

A continuación, se describen los objetivos de esta actividad.

#### Objetivo General


- Desarrollar un sistema que permita la gestión de la base de datos del banco

#### Objetivos específicos


- Realizar la correcta distribución de la base de datos
- Desarrollar cada una de las APIs requeridas para la manipulación de las distribuciones de la base de datos
- Ejecutar las pruebas requeridas a las APIs

### Software utilizado

A continuación, se describe cada una de las herramientas utilizadas para el correcto desarrollo de esta actividad.

Icono	Nombre	Descripción
	MySQL 5.7	Sistema de gestión de bases de datos relacionales comercializado por Oracle contando con un sistema de licencia dual (publica general y comercial). Es uno de los SGBD relacionales más usados en el ámbito



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------



Workbench 8.0

de desarrollo de software debido a su facilidad de uso.

Cliente grafico para la gestión y manipulación de bases de datos MySQL Presentando la facilidad de crear las bases de datos en base a Modelos de entidades, definiendo de forma gráfica los atributos, entidades y relaciones requeridas.



Java 1.8

Paquete de desarrollo para el lenguaje de Java comercializado por Oracle, siendo la versión 1.8 la ultima y más estable de este y con la cual el correcto desarrollo de productos de software bajo el lenguaje Java.



Eclipse IDE for  
Enterprise Java  
Developers 2020-  
09

Entorno de desarrollo para Java que permite el desarrollo de programas en dicho lenguaje y distintos productos de software, permitiendo la instalación de componentes de desarrollo de distintos ámbitos.




Spring Boot 4

Framework para el desarrollo de aplicaciones web en lenguaje Java, simplificando el proceso del desarrollo enfocado en el apartado de bases de datos.



Postman

Herramienta para el envío de peticiones HTTP a servidores, usada con el fin de poder probar el correcto funcionamiento de servidores sin tener aun el cliente final de estos.

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------



Github

Plataforma de desarrollo colaborativa que presenta facilidades en el alojamiento de proyectos mediante el control de versiones.



Github Desktop

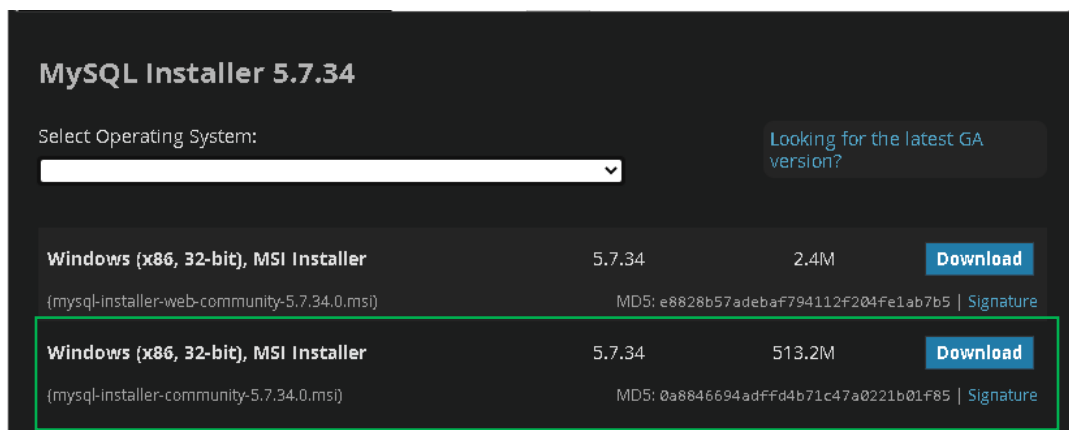
Programa de escritorio que permite la manipulación de repositorios Git. Permite la clonación de los repositorios y el control de las versiones de los repositorios.

### Instalación de software


Para el correcto desarrollo de esta actividad debemos contemplar la instalación o adecuación de todo el software previamente expuesto, lo cual se explicará a continuación:

#### Instalación de MySQL 5.7

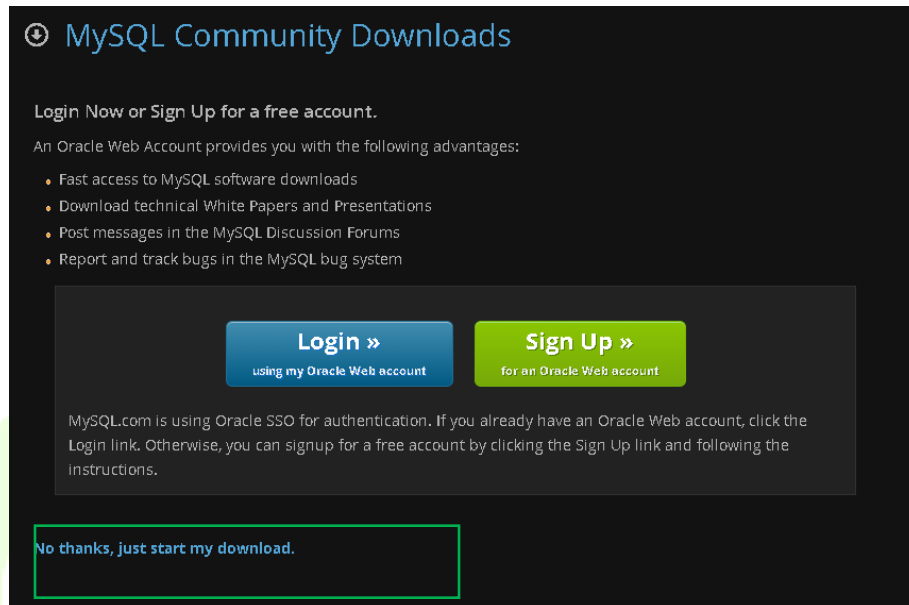
Para la descarga del archivo de instalación de MySQL 5.7, nos dirigiremos al siguiente enlace <https://dev.mysql.com/downloads/windows/installer/5.7.html> donde encontraremos dos tipos de archivos:




El primer archivo de instalación corresponde al que permite la instalación web de MySQL, en nuestro caso, haremos uso del segundo archivo el cual contiene todo lo

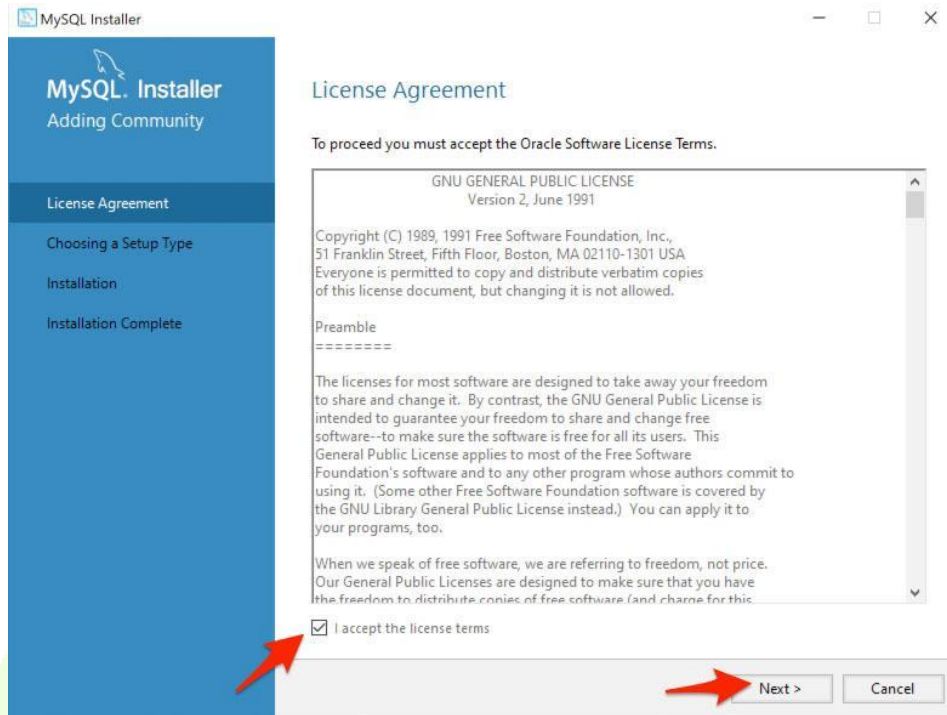
 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

requerido para la instalación de MySQL. Para esto daremos clic en el botón de Dowload de ese archivo, el cual nos remitirá a la pagina de validación de MySQL Community Downloads, en esta página, nos saltaremos la validación de usuario, dando clic en **No thanks, just start my download.** (También puede dar clic en el enalce anterior para iniciar la descarga directamente).




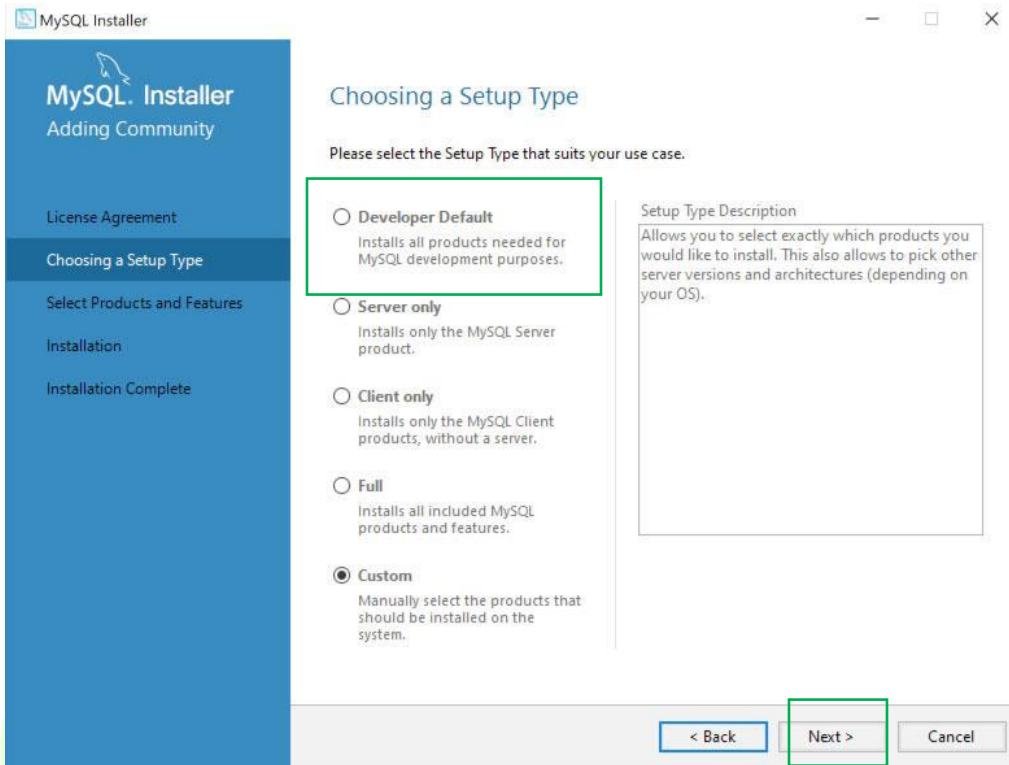
Al finalizar la descarga del archivo, lo abriremos dándole permisos de administrador (Clic derecho sobre el archivo y clic en ejecutar como Administrador) y nos encontraremos con la Licencia, la cual le daremos clic en **I accept the license terms.**

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------




Luego nos enviara al panel en donde seleccionaremos que tipo de instalación requerimos, ya que requerimos usar a MySQL para el desarrollo, elegiremos la opción **Developer Default** y daremos clic en **next**

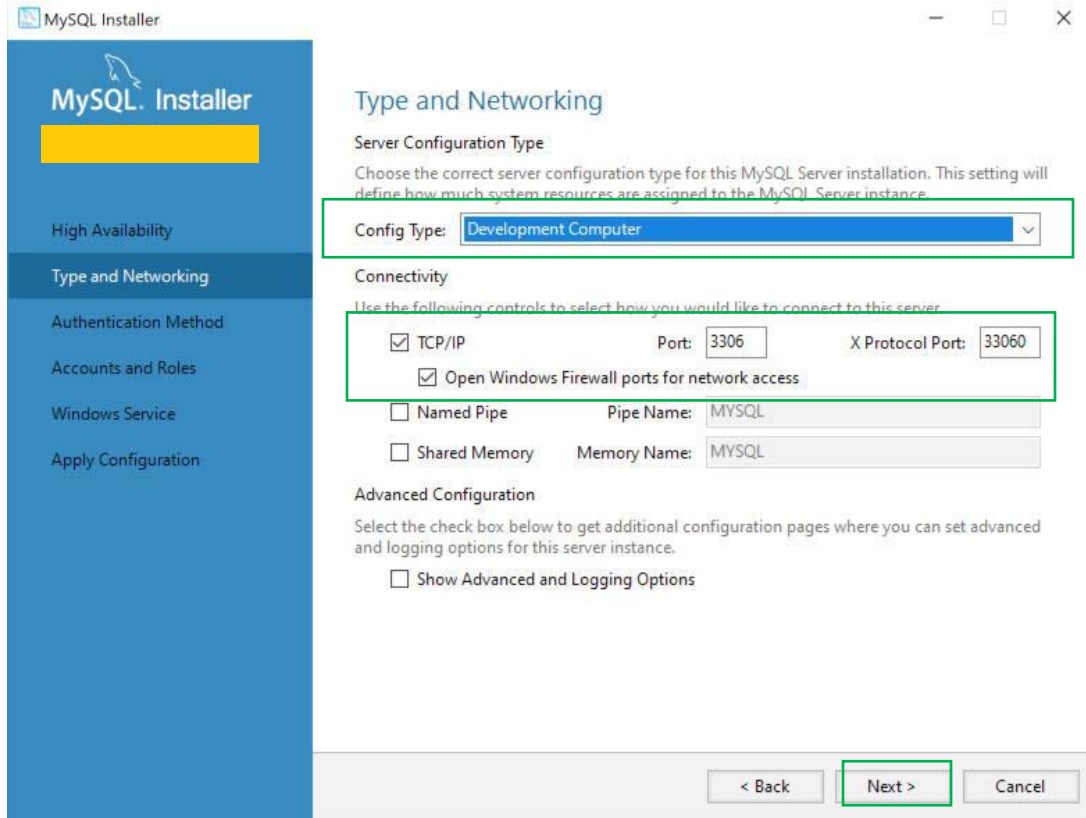
 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------



A continuación, se iniciará el proceso de instalación de todos los componentes incluidos en este tipo de instalación, además de instalar las dependencias o componentes extra requeridos para el correcto funcionamiento.


Después de la instalación de todas las dependencias, componentes y conectores de MySQL, configuraremos el servidor:

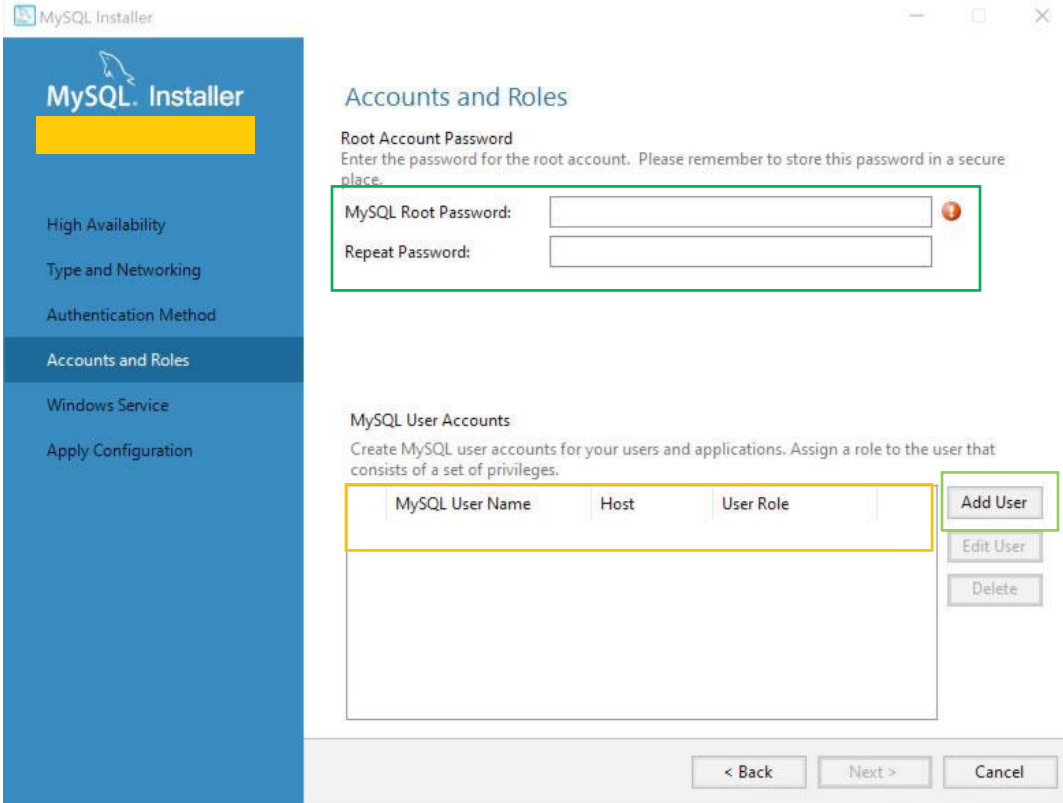
 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
--	---	--	----------




Dejaremos como Config Type a Development Computer y el puerto 3306, revisando que este seleccionada la opción **Open Windows Firewall ports to network Access**.

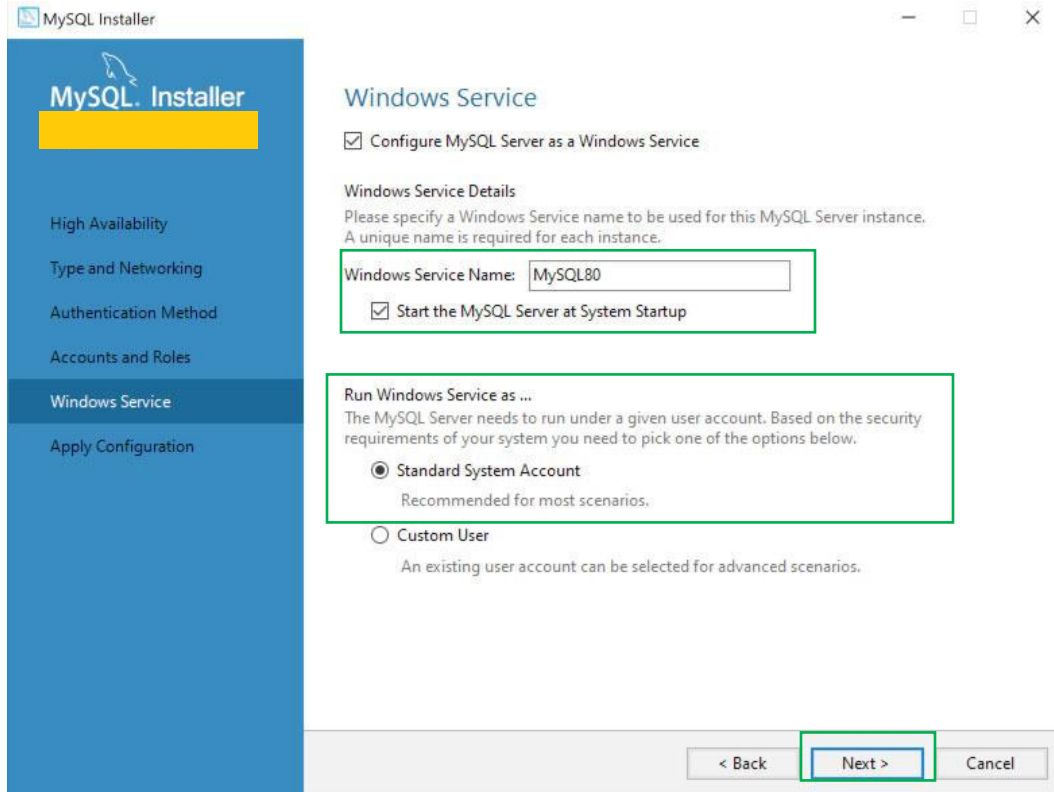
Luego definiremos la contraseña para el usuario **Root**, recuerde esta contraseña ya que será requerida en el proceso de desarrollo. Al introducir y validar la contraseña, le daremos clic en **Add User** para guardar las credenciales del usuario Root en el cuadro de la parte inferior nos mostrara que se almaceno correctamente las credenciales. Al guardarse las credenciales, daremos clic en Next para continuar con el proceso.

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
--	---	---	----------




A continuación, nos remitirá al panel en donde nos mostrará la configuración de MySQL como servicio de Windows, aquí solo verificaremos el nombre brindado por el instalador, que se encuentre seleccionada la opción **Start the MySQL Server at System Startup**, esto con el fin de garantizar que el servicio de MySQL inicie cada vez que encendamos nuestro equipo, también verificaremos el apartado de Run Windows Service as se encuentre definido como **Standard System Account**.

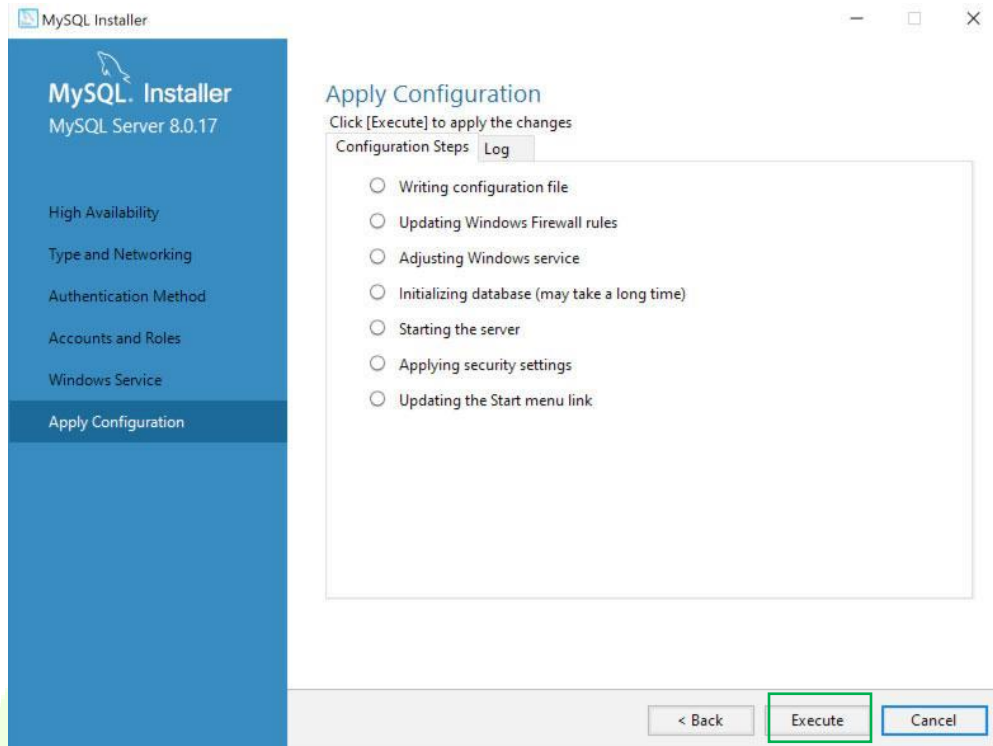
 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------



Al dar clic en Next, nos remitirá al ultimo panel en donde se nos pedirá ejecutar toda la configuración requerida para MySQL, para esto daremos clic en el botón **Execute**, y nos indicará cuando se termine todo el proceso de instalación y configuración.



 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
--	---	---	----------




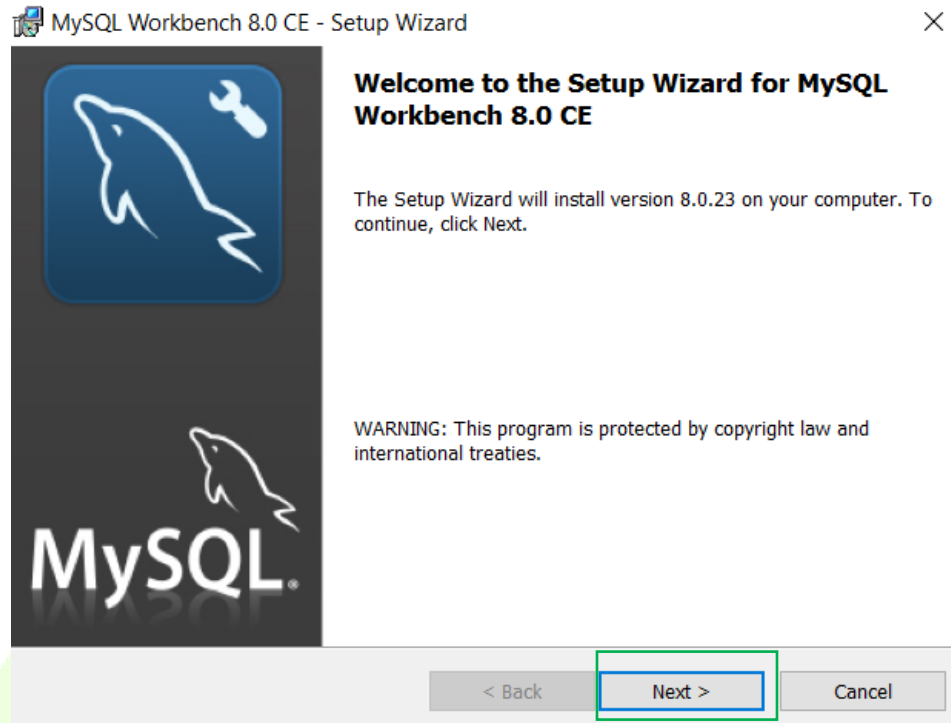
Cuando todos los círculos de cada uno de los ítems estén en verde podremos dar clic en **Finish** para terminar todo el proceso de instalación.

#### Instalación de MySQL Workbench 8.0


Comúnmente podemos obtener a Workbench en la instalación de MySQL, pero, si se da el caso en que omitimos su instalación, solo requerimos dar clic en el siguiente enlace para iniciar la descarga del archivo de instalación <https://dev.mysql.com/get/Downloads/MySQLGUITools/mysql-workbench-community-8.0.25-winx64.msi>.

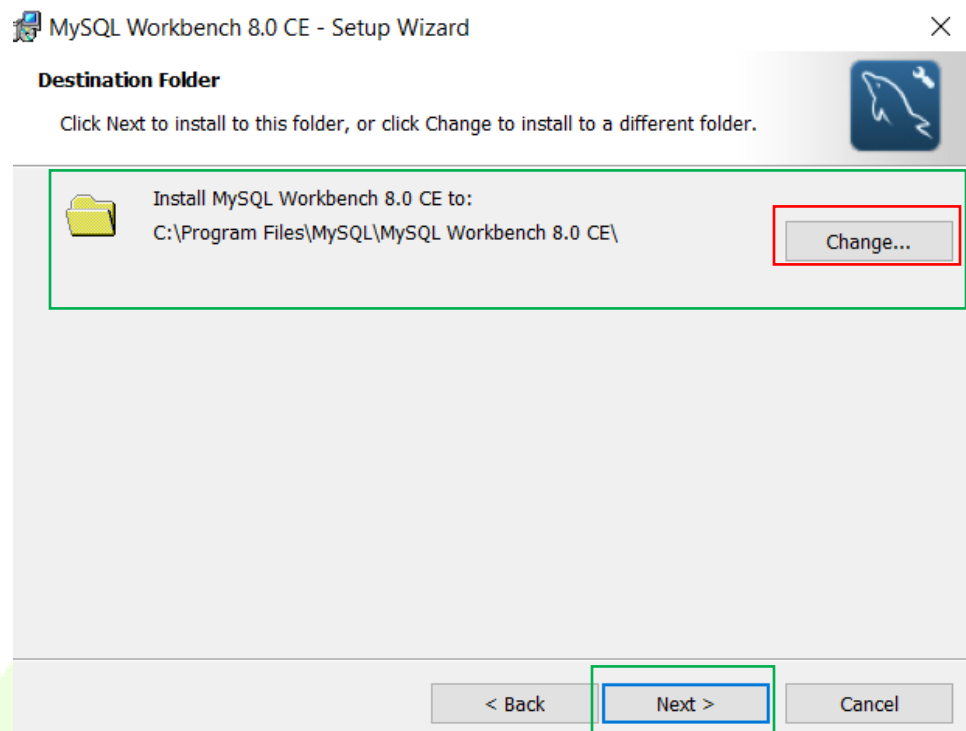
Ya con el archivo descargado, le daremos clic derecho y lo ejecutaremos como administrador. La primera pantalla que nos encontraremos será la siguiente. En esta solo daremos clic en Next para iniciar con la instalación.

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------




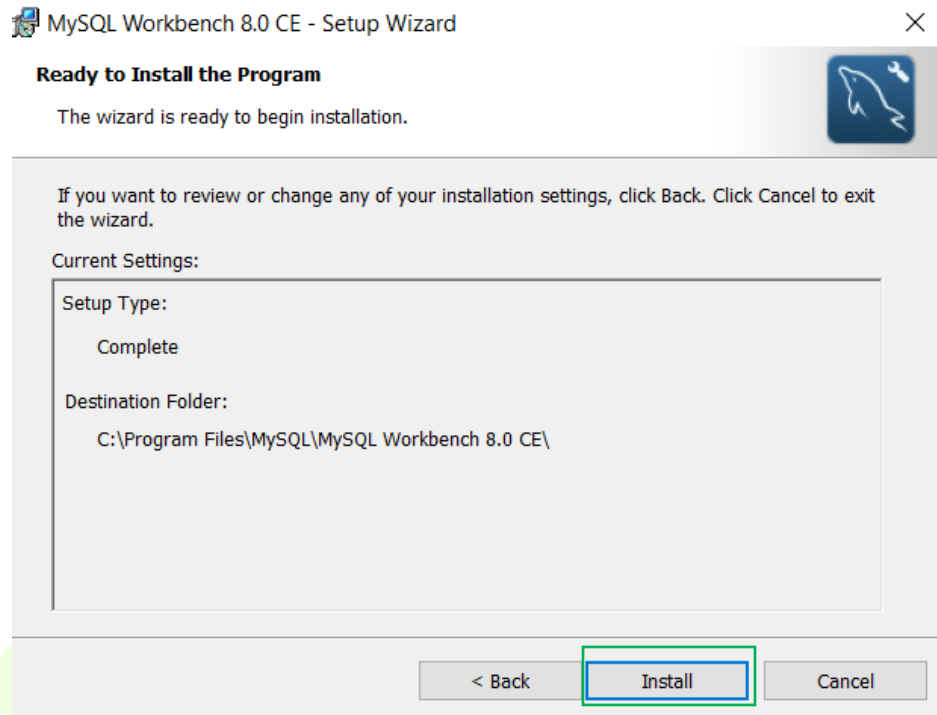
En la siguiente pantalla nos mostrara la ubicación en la que se instalara Workbench, si requiere cambiar la ubicación de clic en el botón **Change**, al finalizar de clic en Next ´para iniciar con el proceso de instalación.

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
--	---	--	----------




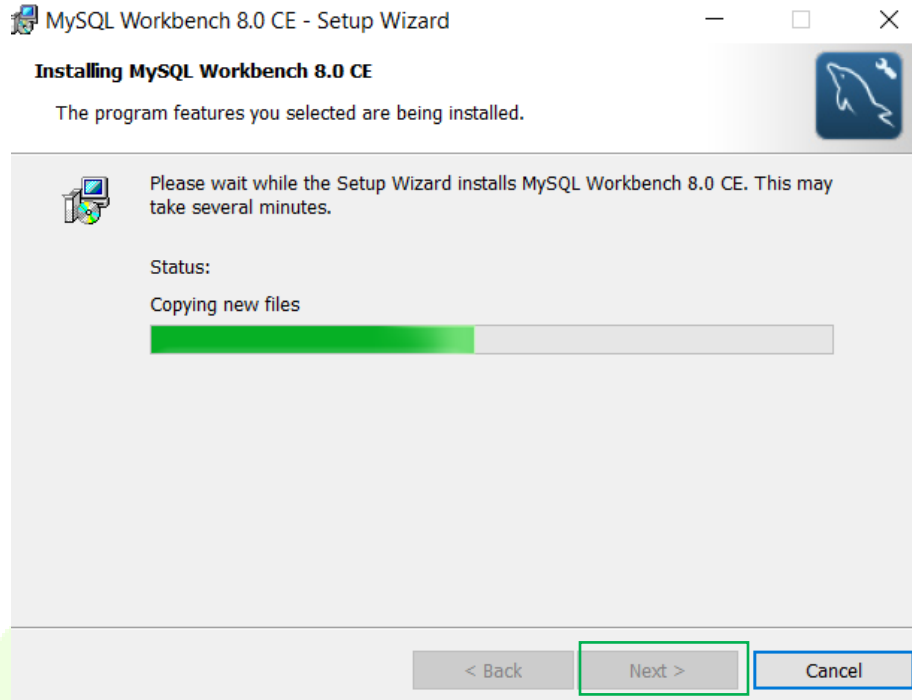
Antes de iniciar con el proceso de instalación se nos mostrara información sobre el proceso de instalación, indicándonos el tipo de instalación y el directorio donde se instalará. AL revisar estos datos, daremos clic en **Install** si estamos de acuerdo.

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
--	---	--	----------




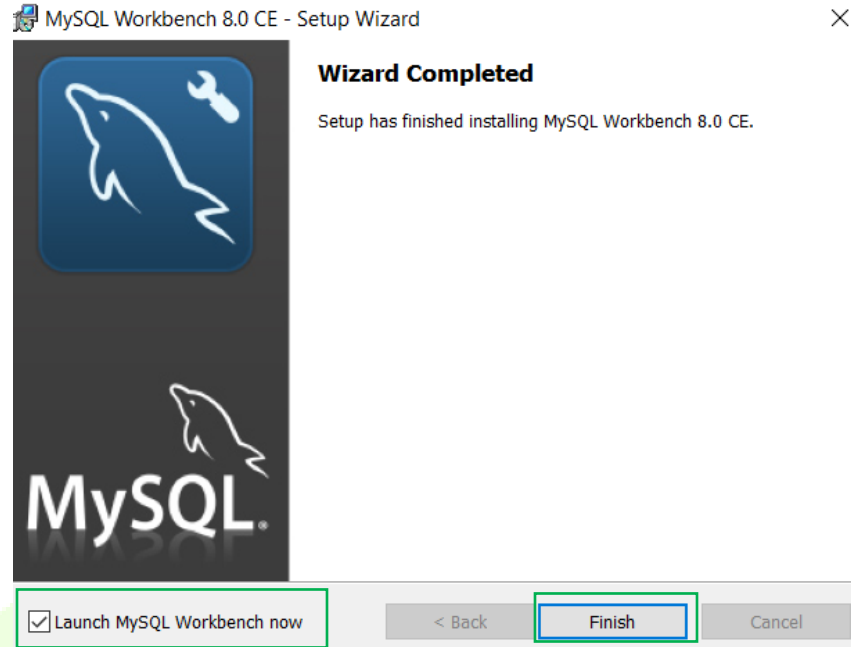
Luego se nos mostrara una barra que indicara el estado del proceso de instalación de Workbench. Al finalizar este proceso daremos clic en el botón **Next** que se habilitara al finalizar el proceso.

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------



Al finalizar, nos indicara que se termino la instalación, si requerimos iniciar con a Workbench, revisaremos que la opción **Launch MySQL Workbench now** esta marcada, si no requerimos que se inicie aun el programa, desmarcaremos esa opción.


 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
--	---	--	----------

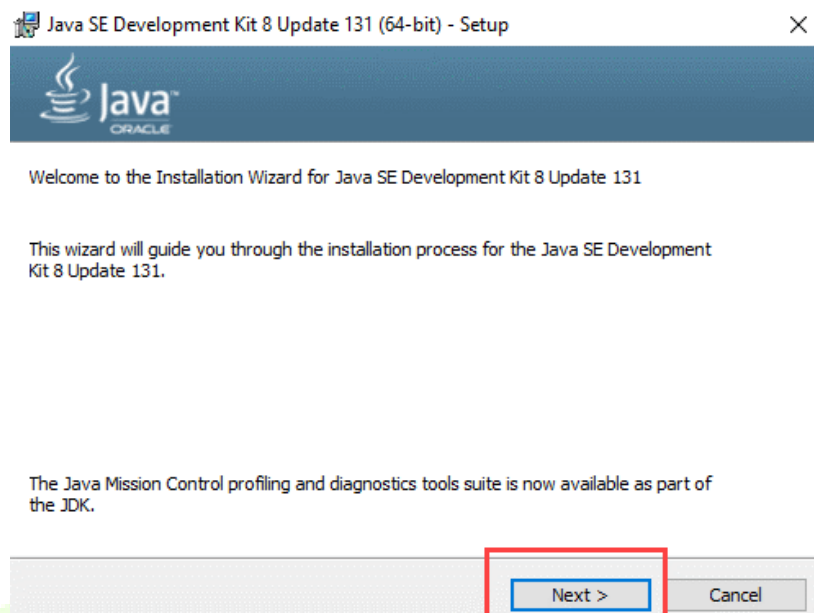


#### Instalación de Java 1.8

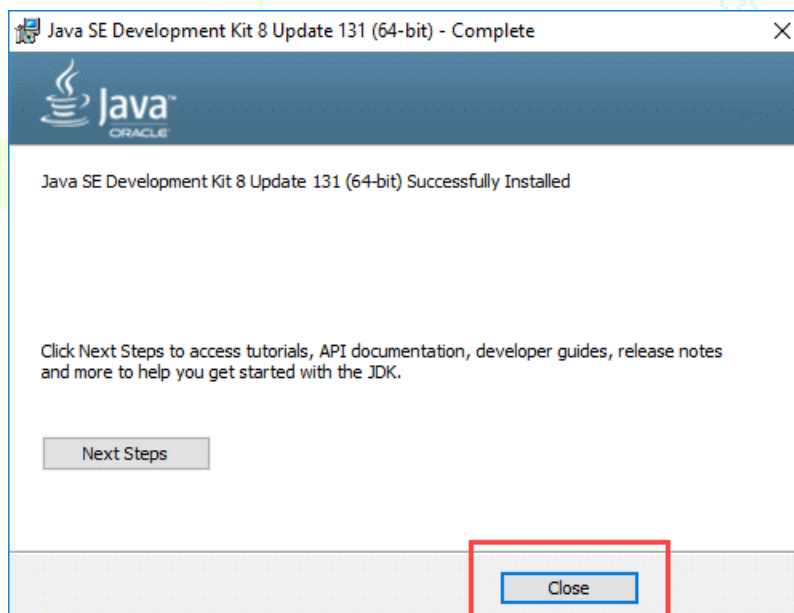
Para la instalación del paquete de desarrollo de Java en su versión 8, daremos clic en este **enlace**, el cual nos remitirá a la pagina de Oracle, dado sea el caso que nos pida que ingresemos con una cuenta de Oracle, si no cuenta con una, solo deberá crearla, y luego podrá realizar la descarga.

Ya al finalizar todo el proceso de descarga, iniciaremos con la instalación del JDK, para esto daremos clic en el archivo ejecutable, el cual, al abrirse nos mostrara la siguiente ventana:

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------




En esta ventana solo daremos clic en Next para iniciar con el proceso de instalación. El cual, al finalizar, nos mostrara la siguiente ventana.



Instalación de Eclipse Enterprise Edition

Para la instalación de Eclipse, en su última versión vigente, nos dirigiremos al siguiente enlace <https://www.eclipse.org/downloads/packages/release/2019->

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------

06/r/eclipse-ide-enterprise-java-developers en donde nos encontraremos con el link de descarga.



**Eclipse IDE for Enterprise Java Developers**

**Package Description**  
Tools for Java developers creating Enterprise Java and Web applications, including a Java IDE, tools for Enterprise Java, JPA, JSF, Mylyn, Maven, Git and more.  
Click [here](#) to file a bug against Eclipse Web Tools Platform.  
Click [here](#) to file a bug against Eclipse Platform.  
Click [here](#) to file a bug against Maven integration for web projects.

**This package includes:**

- Data Tools Platform
- Git Integration for Eclipse
- Eclipse Java Development Tools
- Eclipse Java EE Developer Tools
- JavaScript Development Tools
- Maven Integration for Eclipse
- Mylyn Task List
- Eclipse Plug-in Development Environment
- Eclipse XML Editors and Tools

[Detailed features list](#)

**Download Links**  
Windows x86\_64  
macOS x86\_64  
Linux x86\_64

Downloaded 550,312 Times

[Checksums...](#)

**Bugzilla**  
[Open Bugs: 74](#)  
[Resolved Bugs: 166](#)  
[File a Bug on this Package](#)

**New and Noteworthy**  
[Eclipse Web Tools Platform Project](#)

The Eclipse Installer 2021-03 R now includes a JRE for macOS, Windows and Linux.

**Get Eclipse IDE 2021-03**  
Install your favorite desktop IDE packages.  
[Download x86\\_64](#)  
[Download Packages](#) | [Need Help?](#)

**RELATED LINKS**

- [Compare & Combine Packages](#)
- [New and Noteworthy](#)
- [Install Guide](#)
- [Documentation](#)
- [Updating Eclipse](#)


Maintained by: WTP and the Eclipse Packaging Project

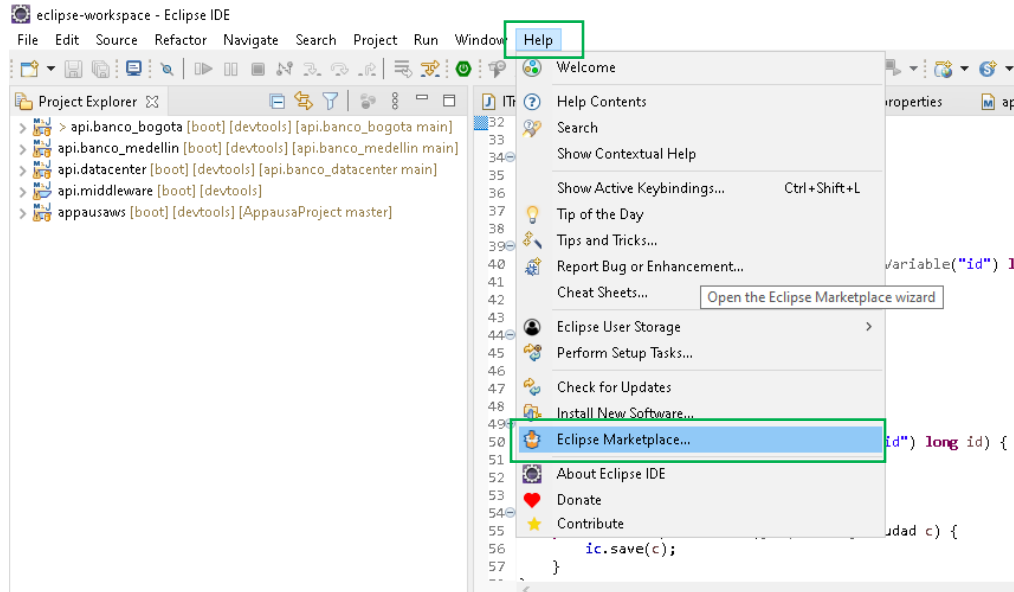
Al descargar el archivo de instalación, lo ejecutaremos dándole los permisos requeridos, y seguiremos los pasos de instalación indicados. El proceso de descarga de todos los archivos requeridos puede ser demorado.

#### Instalación de Spring Boot 4

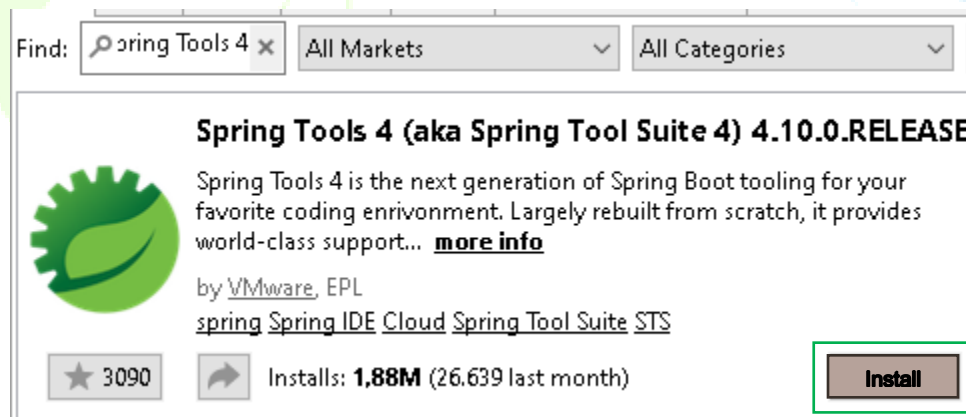
Para instalar Spring Boot y poderlo usar correctamente en Eclipse, debemos abrir a Eclipse, luego nos dirigiremos a la barra superior, abriremos el apartado de **Help** o **Ayuda**, y abriremos el apartado de Eclipse Marketplace.



 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------




Dentro de este apartado, nos encontraremos con la página de búsqueda, aquí escribiremos Spring tools 4, obteniendo el siguiente resultado, al cual le daremos clic en el botón **Install**, el cual nos mostrara el proceso de instalacion:

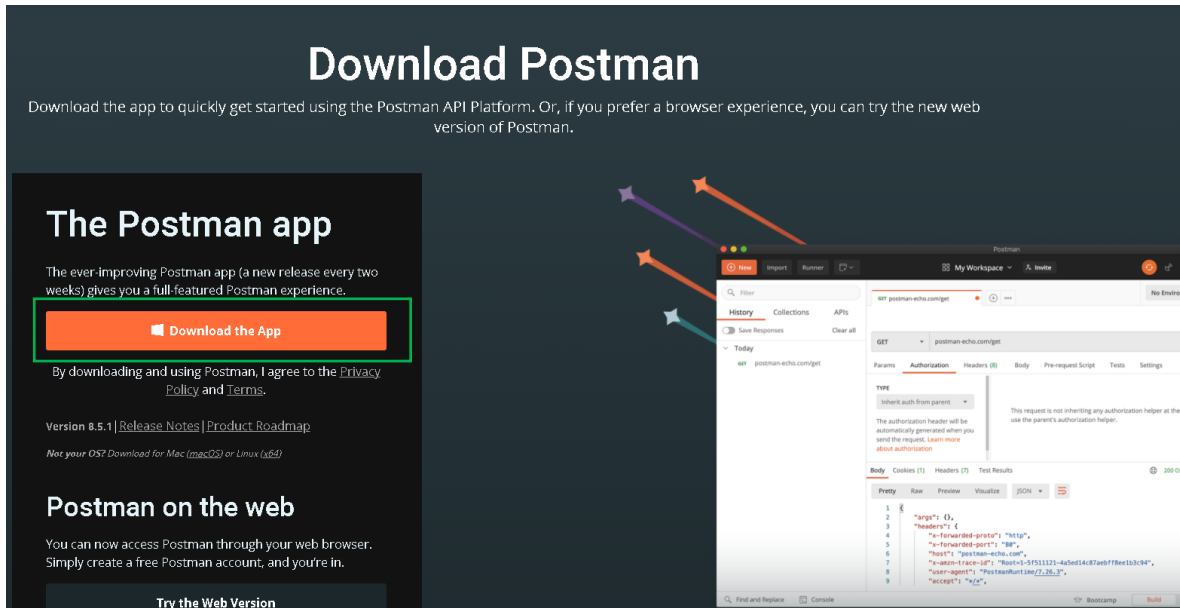


Al finalizar todo el proceso, se nos pedirá reiniciar a Eclipse, en donde ya encontraremos a Spring Boot listo para su uso.

#### Instalación de Postman

Para la instalación de Postman, programa por medio del cual podremos realizar las pruebas de las APIs, nos dirigiremos al siguiente enlace <https://www.postman.com/downloads/>, aquí daremos clic en el botón indicado para descargar el archivo de instalación.

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------



**Download Postman**

Download the app to quickly get started using the Postman API Platform. Or, if you prefer a browser experience, you can try the new web version of Postman.

**The Postman app**

The ever-improving Postman app (a new release every two weeks) gives you a full-featured Postman experience.

[Download the App](#)

By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

Version 8.5.1 | [Release Notes](#) | [Product Roadmap](#)

Not your OS? Download for [Mac \(macOS\)](#) or [Linux \(x64\)](#)

**Postman on the web**

You can now access Postman through your web browser. Simply create a free Postman account, and you're in.

[Try the Web Version](#)

Postman interface showing a GET request to `postman-echo.com/get` with headers:


```

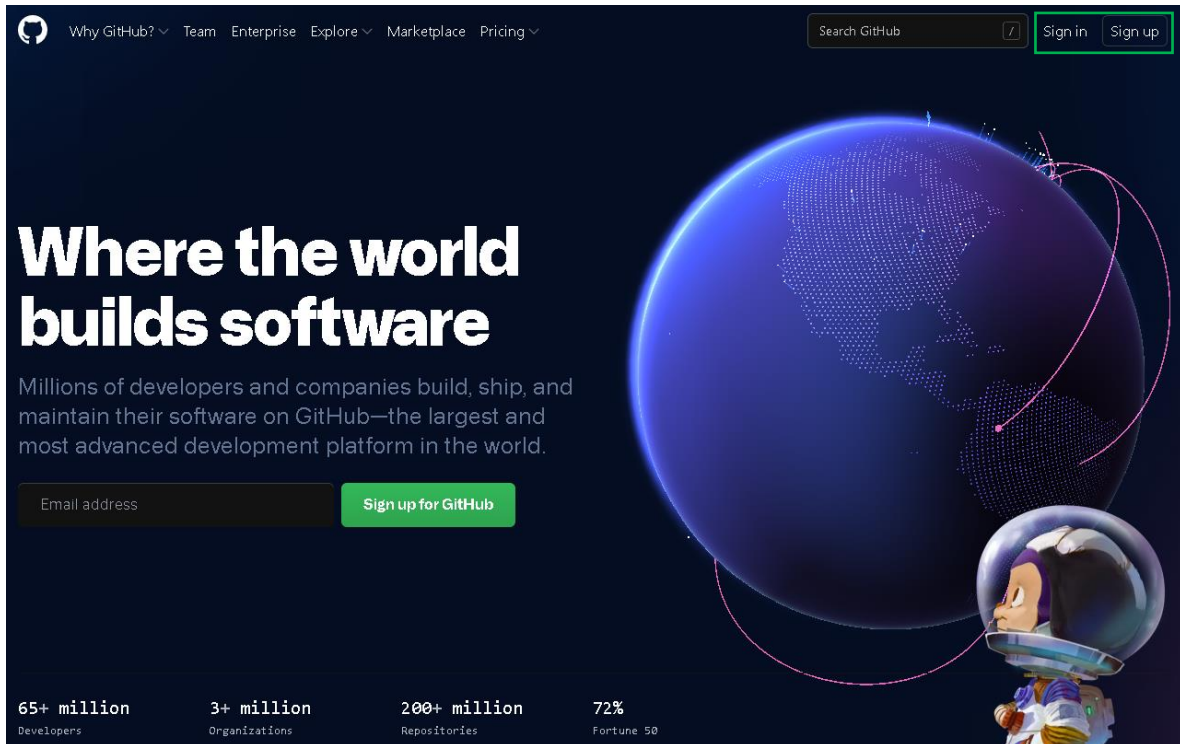
{
  "args": {},
  "headers": {
    "x-forwarded-port": "8080",
    "x-forwarded-port": "8080",
    "host": "postman-echo.com",
    "x-api-token": "Basic SF531323-4d5014c87a0f7f8e13334",
    "user-agent": "PostmanRuntime/7.28.2",
    "accept": "*/"
  }
}
```

Al finalizar la descarga, podemos iniciar el proceso de instalación, sin ninguna complicación o requerimiento extra, siguiendo las instrucciones que nos indica el instalador. Al finalizar el proceso de instalación, se nos pedirá tener una cuenta de Postman para hacer uso del programa (inicie sesión con una cuenta existente o cree una nueva dado sea el caso que lo requiera)

### Cuenta de Github

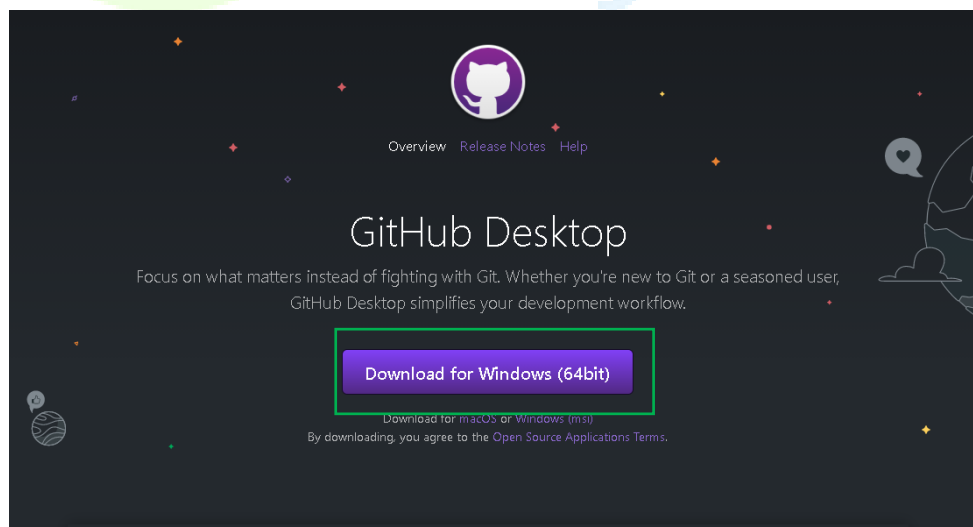
Se requiere que usted cuente con una cuenta de Github para la creación de los repositorios requeridos para el almacenamiento de las APIs, para esto, diríjase al siguiente enlace <https://github.com/> para crear o iniciar sesión con una cuenta de Git.


 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------



### Instalación de GitHub Desktop

Para la instalación de GitHub Desktop, nos dirigiremos al siguiente enlace <https://desktop.github.com/> en donde daremos clic en el botón indicado para descargar el archivo de instalación de este programa.



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

Luego de finalizar la descarga, ejecutaremos el archivo sin ningún problema, siguiendo los pasos indicados por el instalador. Luego de tener ya instalado el programa, iniciaremos sesión con nuestra cuenta de GitHub.

## Desarrollo

En esta sección se describirá todas las actividades realizadas para el desarrollo de la actividad, contemplando desde la creación de la base de datos, de los repositorios y las APIS, además de

### Distribución de la base de datos

A continuación, se va a detallar el proceso relacionado con la distribución de la base de datos, incluyendo el análisis y las actividades de algebra relacional que permitieron la distribución acorde a las necesidades expuestas por el cliente.

### Análisis

Un banco desea distribuir su base de datos en dos puntos con sistemas de bases de datos homogéneos, los cuales se deben coordinar por medio de un tercer punto, el cual está encargado de realizar periódicamente un proceso de sincronización de los datos de ambos nodos.

Los puntos resultantes se encuentran en dos ciudades diferentes, en este caso Bogotá y Medellín. En cada nodo se almacenarán los datos de las transacciones de las cuentas registradas en ellas, es decir, si una cuenta es registrada en Bogotá, los datos de esa cuenta se almacenarán en el nodo de Bogotá. En el punto de coordinación se deben registrar en la tabla de auditoria todas las transacciones realizadas por los usuarios.

Cada nodo necesita tener un administrador encargado de registrar, acceder y eliminar los datos que se almacenen en ellos. El cuerpo laboral de cada sede del banco (nodo) puede acceder a los datos de las cuentas de sus clientes y registrar nuevas cuentas.

### Objetivos


Para la distribución adecuada de la base de datos se deben establecer ciertas pautas que definirán los lineamientos a seguir para el desarrollo del proyecto.

#### *Objetivo General*

- Distribuir los datos de las cuentas de los clientes y las transacciones que se realicen en ellas sobre los nodos respectivos.

#### *Objetivos específicos*

- Crear las bases de datos en donde se alojarán los datos.


 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

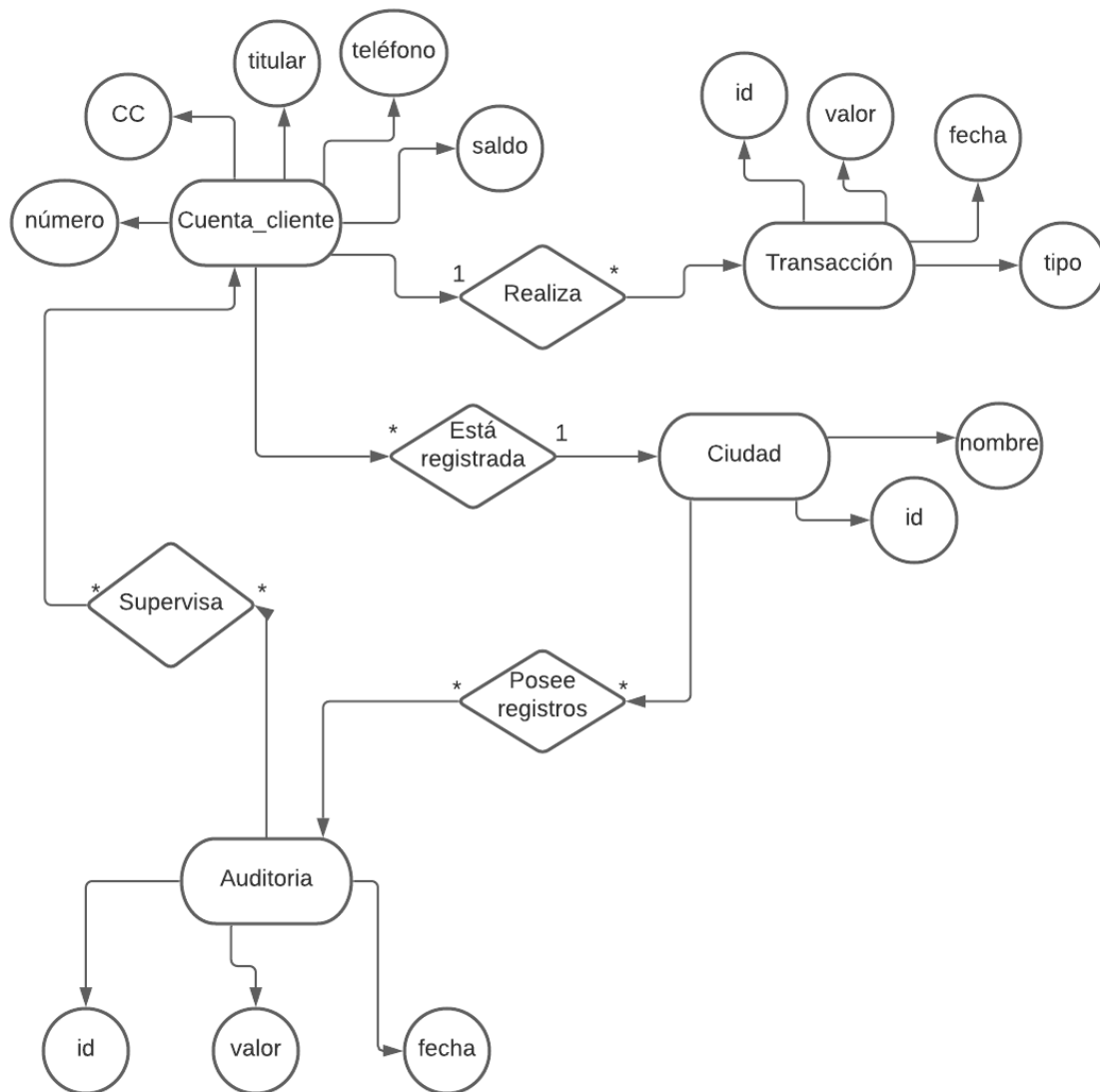
- Definir el tipo de fragmentación a implementar.
- Detallar el procedimiento para la fragmentación de las tablas.
- Fragmentar las tablas de la base de datos.
- Almacenar los fragmentos en las bases de datos correspondientes.

#### Diseño conceptual

A continuación, se describe el diseño conceptual de la base de datos requerida por el banco, especificando las actividades y relaciones que tienen las entidades descritas por el cliente.




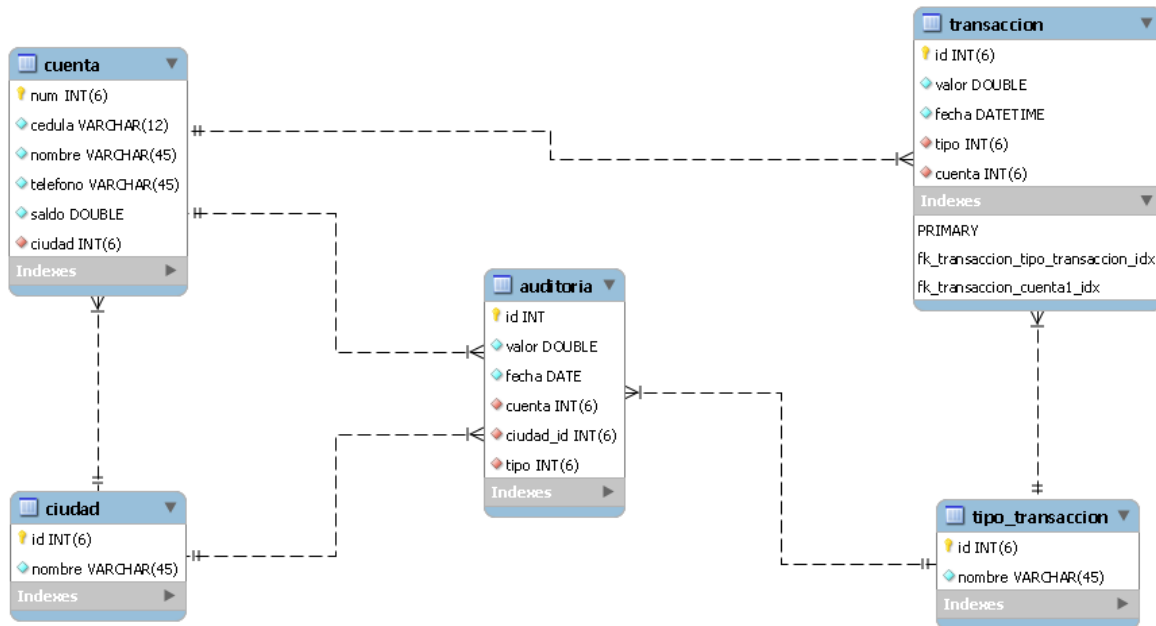
 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------



### Esquema conceptual global

En base al diseño conceptual realizado se realiza el esquema conceptual definiendo las relaciones, tipos de datos requeridos por cada tabla, aun sin contemplar el tipo de distribución requerida por el cliente, pensando en desplegar posteriormente la base de datos de forma centralizada, para realizar el análisis y distribución requerida.

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------



### Diseño de las vistas


En este apartado se definirán las interfaces que utilizan los usuarios finales para acceder a la información almacenada en las bases de datos. Las interfaces comprenden a: los usuarios que interactúan con los nodos, los privilegios que tienen con respecto a su nivel de acceso en la base de datos, la información a la cuál van a acceder y los dispositivos, programas y/o mecanismos desde donde van a acceder.

### Usuarios

Los usuarios que interactúan con el sistema son:

- **Administrador:** Usuario encargado de registrar los datos de las cuentas en su respectiva sede. Puede acceder, modificar y eliminar estos datos cuando sea necesario hacerlo.
- **Cuerpo laboral (Asesores/Gerentes/Cajeros):** Usuario encargado de atender a los clientes del banco. Pueden acceder y modificar los datos de las cuentas, además de registrar nuevas cuentas de ser necesario.

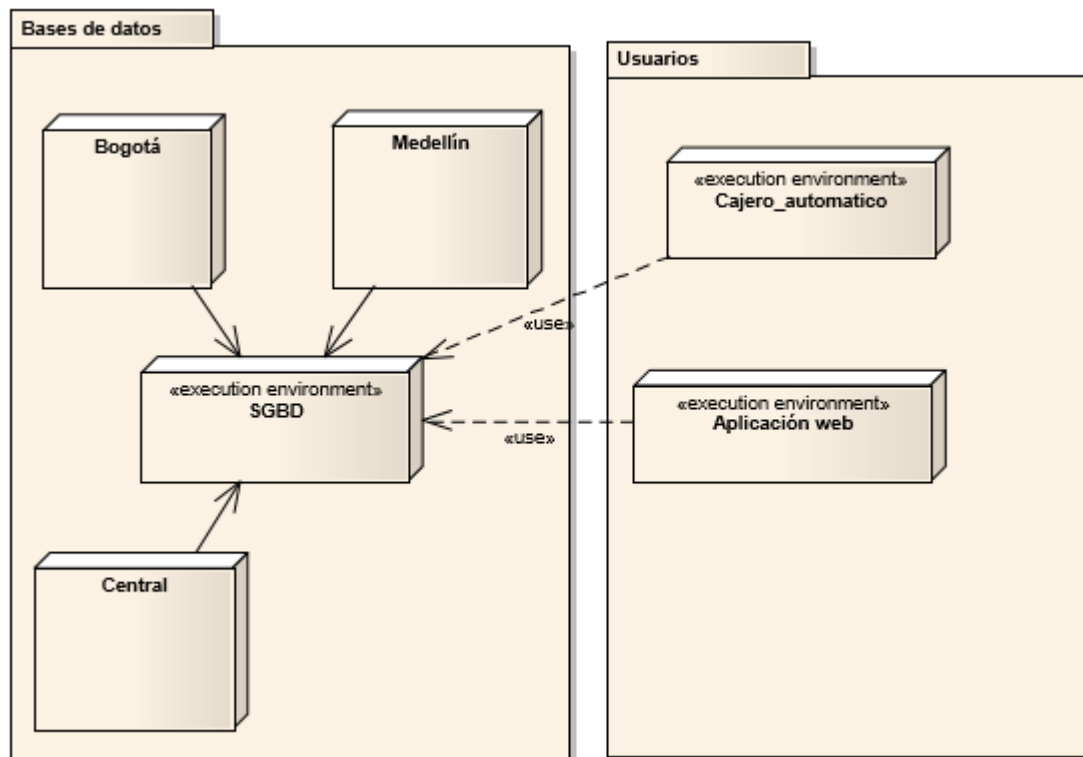
Cada nodo debe estar gestionado por un administrador, pero sólo los nodos de las ciudades poseen un cuerpo laboral encargado de realizar sus funciones respectivas dentro del banco.

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
--	---	--	----------

### Información de acceso

Tablas	Usuarios	Acciones: (R)egistrar, (L)eer, (M)odificar, (E)liminar.
Cuenta_cliente	Administrador, Cuerpo laboral	RELM, RLM
Transaccion	Administrador, Cuerpo laboral	RELM, RLM
Ciudad	Administrador	RELM
Tipo_transacciones	Administrador, Cuerpo laboral	RELM, RLM
Auditoria	Administrador	RELM


### Definición de esquemas externos



### Diseño de la distribución

En base a los datos suministrados, se fragmentará la base de datos de forma horizontal en determinadas tablas basándonos en el dato de ciudad de las cuentas



 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

de los clientes, acogiéndonos a que las transacciones de las cuentas se deben almacenar en la ciudad donde el cliente creó la cuenta.

De modo tal se fragmentarán las tablas de la siguiente forma:

- Cuentas Clientes: esta tabla se fragmentará en base al dato de ciudad de la cuenta, almacenándose en la base de datos correspondiente a esta, además, para facilitar el acceso al saldo de la cuenta, se almacenarán los datos de número de cuenta, saldo y ciudad en que se creó la cuenta, esto último con el fin de facilitar el acceso a esta información por parte de la tabla de auditoria.
- Transacción: estas tablas se fragmentarán bajo la lógica de que se almacenen los datos de las transacciones en la misma ciudad en donde el cliente creó la cuenta, sin importar que este realice transacciones en otra ciudad.

Las tablas de auditoria y tipo de transacción se quedarán en la base de datos central de modo tal la tabla de auditoria las tenga al alcance, además de que se replicara la tabla de tipo de transacción en las otras bases de datos para que estas tengan acceso a la información almacenada. Finalmente, la tabla de Ciudad se replicará en todas las bases de datos, de modo tal se puedan almacenar de forma correcta la ciudad en que se realicen las transacciones foráneas de cada cuenta, además de que facilitará el acceso a la información de las ciudades por parte de la tabla de auditorías.

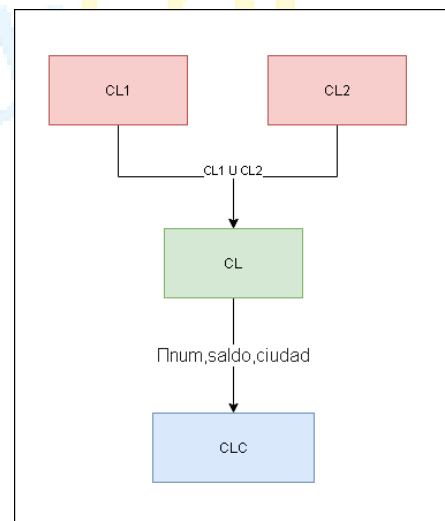
#### *Cuentas clientes*


$CL1 = \rho_{CL1}(\sigma_{ciudad = '101'}(Cuenta))$

$CL2 = \rho_{CL2}(\sigma_{ciudad = '102'}(Cuenta))$

$CLC = \rho_{CLC}(\prod num, saldo, ciudad(Cuenta))$

$CL = \rho_{CL}(CL1 \cup CL2)$



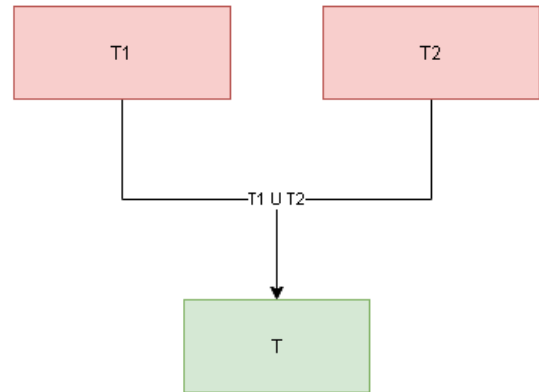
 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------

### Transacción

$TC = \rho TC (Transaccion \times Cuenta)$

$TC1 = \rho TC1 (\sigma Transaccion.cuenta = Cuenta.num \wedge Cuenta.ciudad = '101'(TC))$

$TC2 = \rho TC2 (\sigma Transaccion.cuenta$   
 $= Cuenta.num$   
 $\wedge Cuenta.ciudad = '102'(TC))$



$T1$   
 $= \rho T1 (\Pi Transaccion.id, transaccion.valor, transaccion.fecha, transaccion.cuenta,$   
 $transaccion.tipo (TC1))$

$T2$   
 $= \rho T2 (\Pi Transaccion.id, transaccion.valor, transaccion.fecha, transaccion.cuenta,$   
 $transaccion.tipo (TC2))$


$T = \rho T (T1 U T2)$

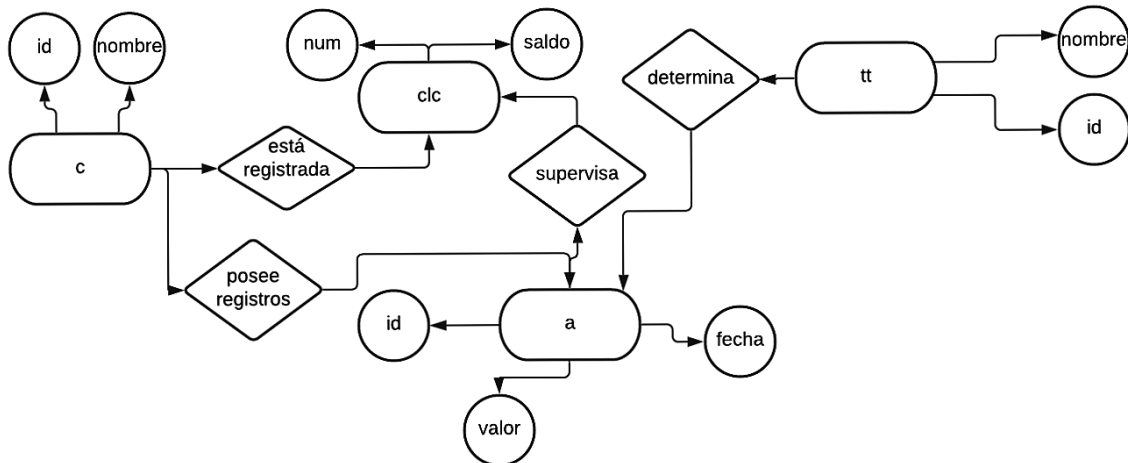
### Esquemas conceptuales locales

En base a la distribución implementada previamente, se crearán los esquemas conceptuales para cada una de las bases de datos resultantes del proceso de fragmentación.

### BD Central

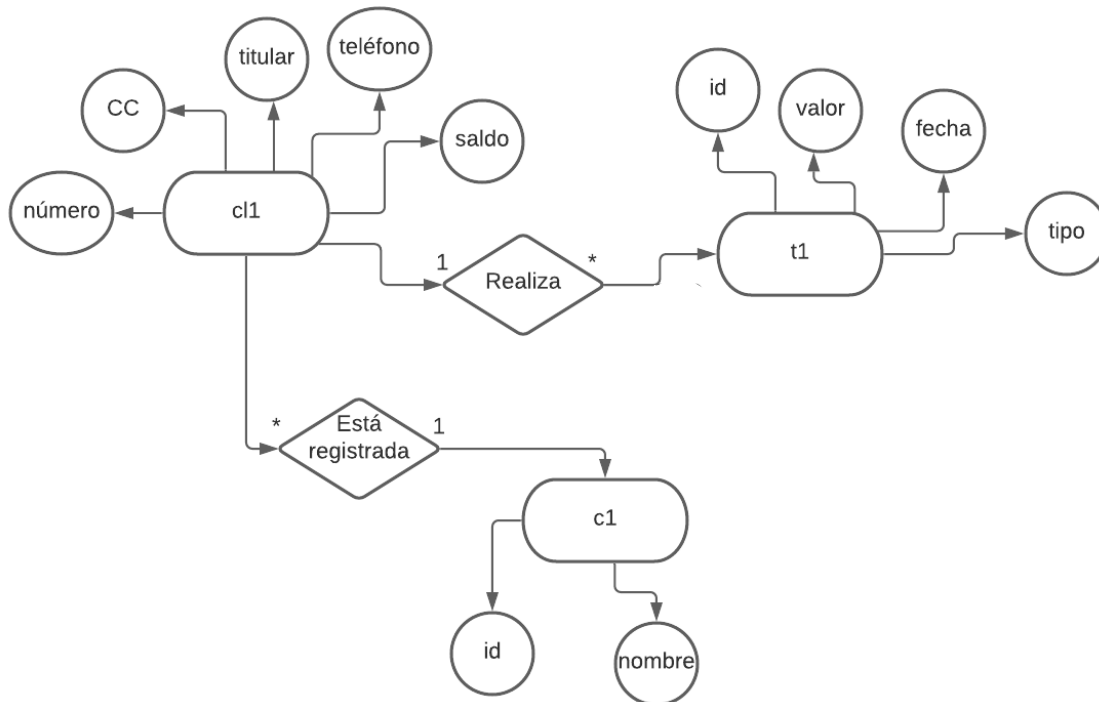
Esta será la base de datos que se implantará en el datacenter del banco, en donde se contemplará el proceso de auditorías.

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------




### BD Bogotá

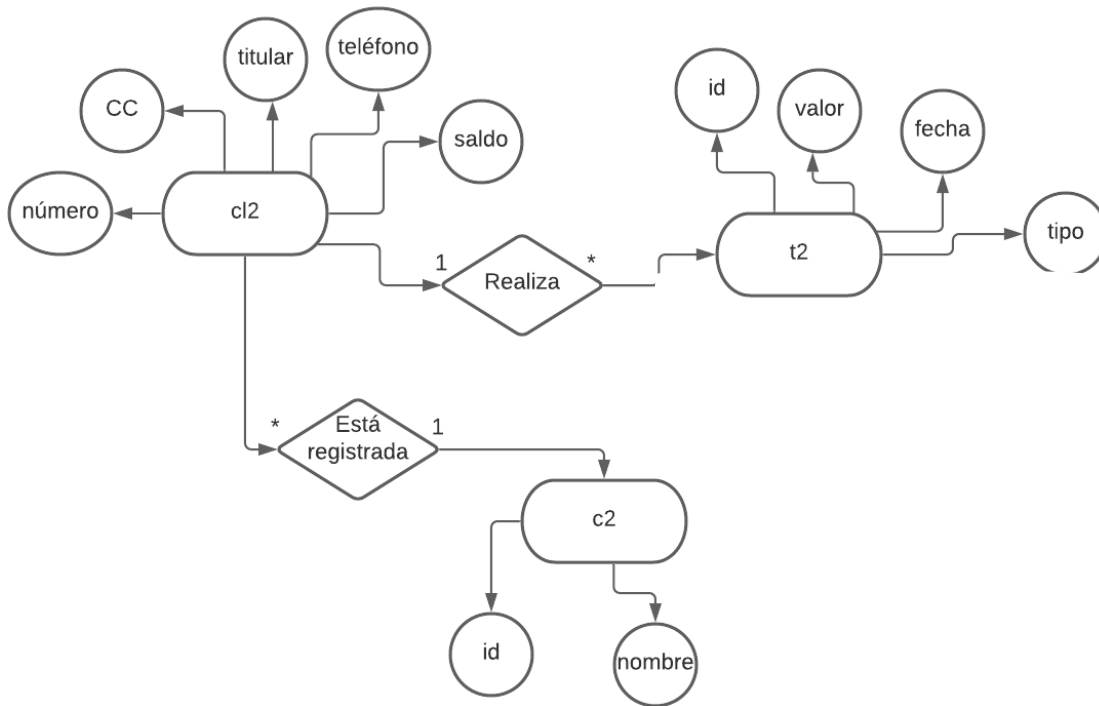
Esta será la base de datos que será implantada en la sede del banco en Bogotá.



### BD Medellín

Esta será la base de datos que será implantada en la sede del banco en Medellín.

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------




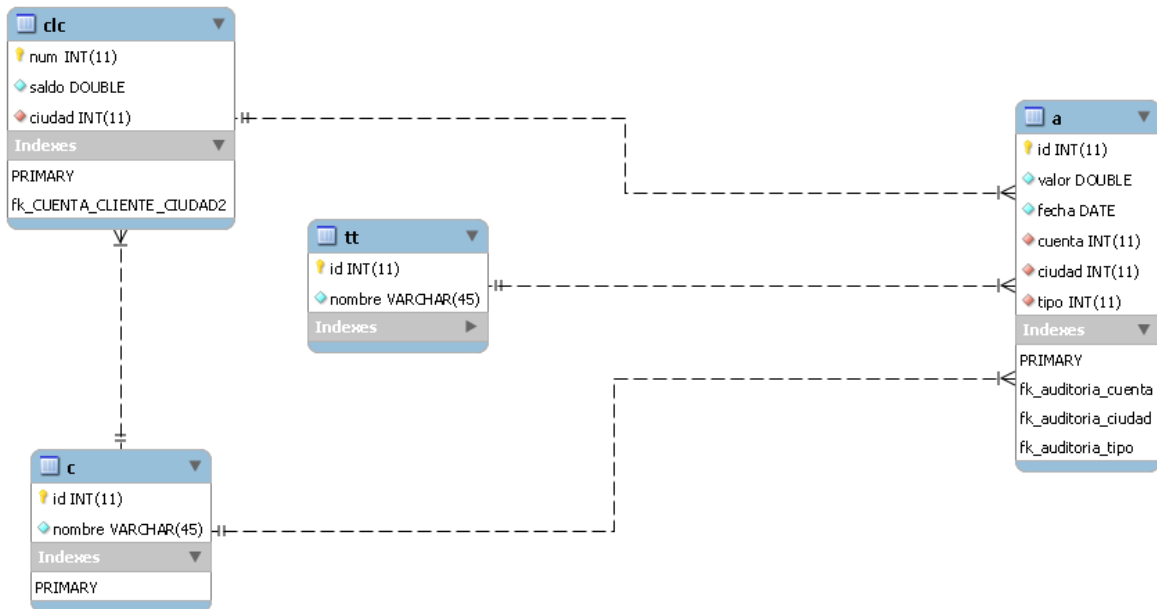
### Diseño físico

Con los esquemas conceptuales realizados, se traducirán en los diseños físicos para cada una de las bases de datos resultantes.

### BD central

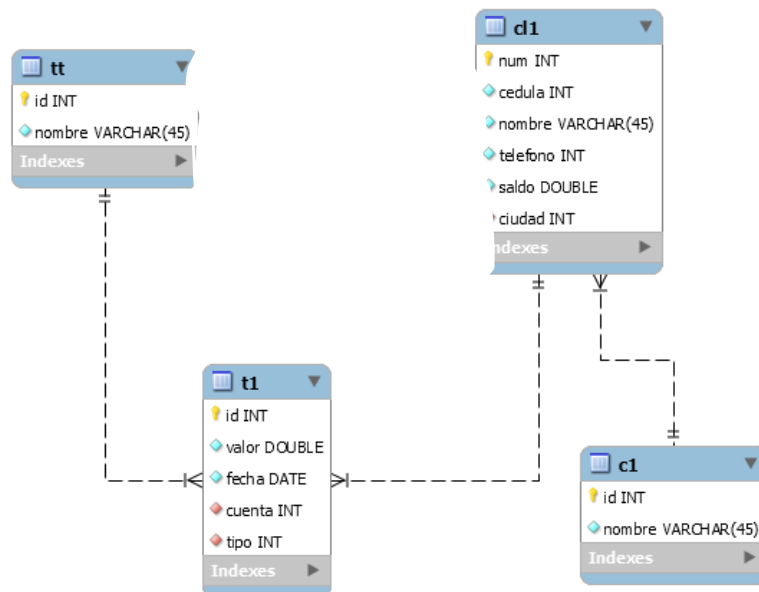
Esta será la base de datos que se implantará en el datacenter del banco.

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------




### BD Bogotá

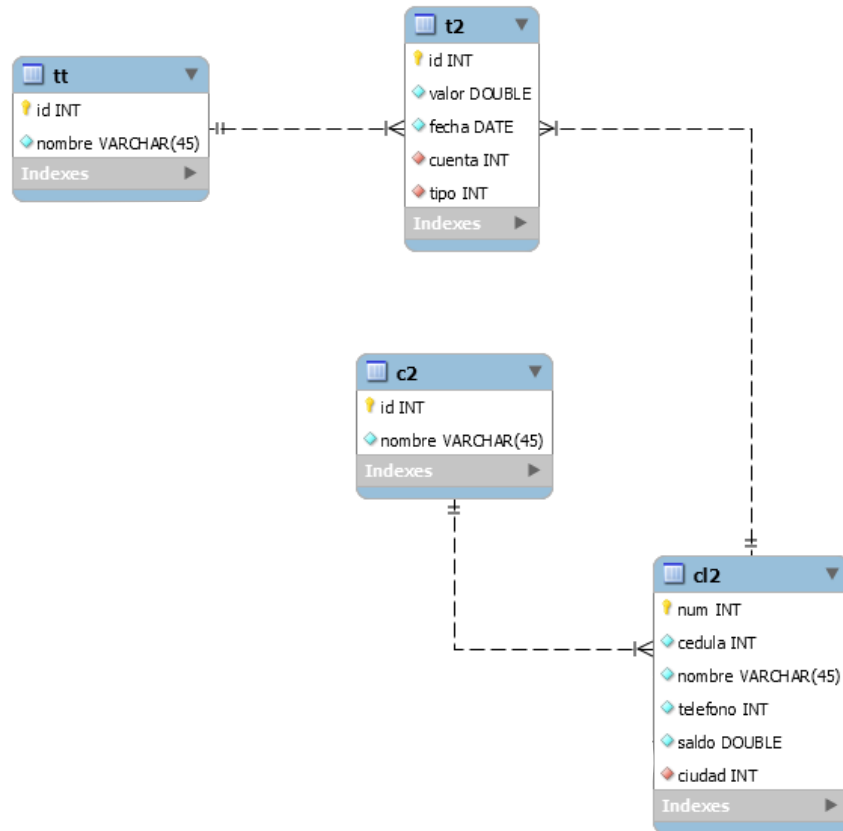
Esta será la base de datos que será implantada en la sede del banco en Bogotá.



### BD Medellín


Esta será la base de datos que será implantada en la sede del banco en Medellín.

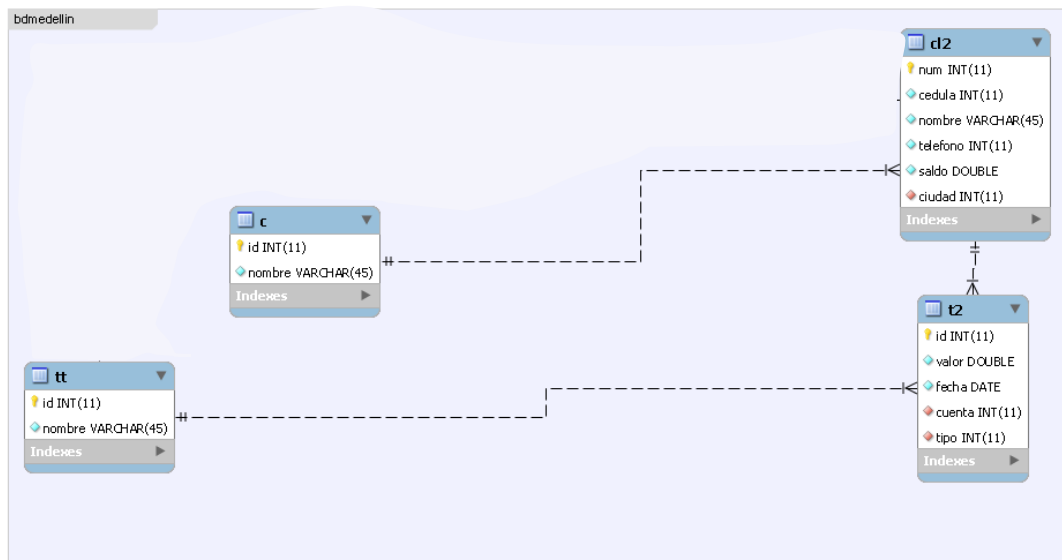
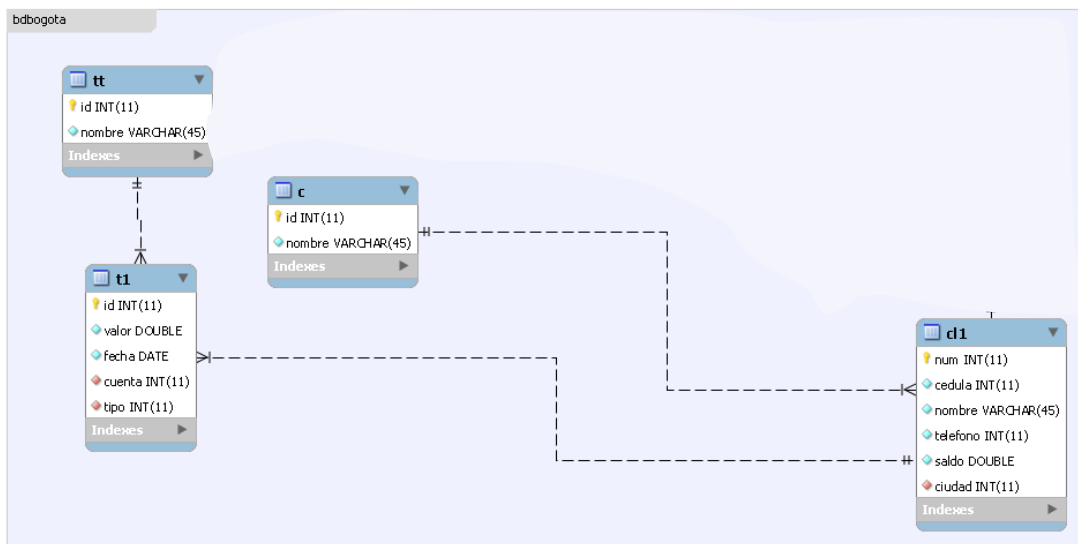
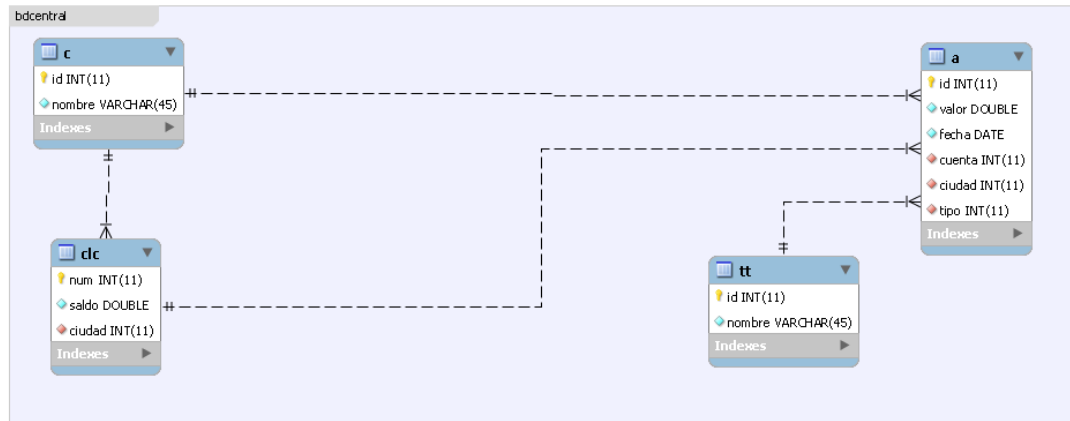
 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------




*Esquema físico*

Finalmente se describen todos los modelos físicos de la base de datos del banco.

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------



 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

### Creación base de datos

Con base a los esquemas físicos resultantes de todo el proceso de distribución de la base de datos del banco, se obtuvieron las siguientes sentencias DDL para crear cada una de las bases de datos.

#### *Banco Bogotá DDL*


```
CREATE SCHEMA IF NOT EXISTS `banco_bogota` DEFAULT
CHARACTER SET utf8 ;
```

```
CREATE TABLE IF NOT EXISTS `banco_bogota`.`ciudad` (
  `id` INT(6) NOT NULL,
  `nombre` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `banco_bogota`.`tipo_transaccion` (
  `id` INT(6) NOT NULL,
  `nombre` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `banco_bogota`.`cuenta` (
  `num` INT(6) NOT NULL,
  `cedula` VARCHAR(12) NOT NULL,
  `nombre` VARCHAR(45) NOT NULL,
  `telefono` VARCHAR(45) NOT NULL,
  `saldo` DOUBLE NOT NULL,
  `ciudad` INT(6) NOT NULL,
  PRIMARY KEY (`num`),
  INDEX `fk_cuenta_ciudad1_idx` (`ciudad` ASC) ,
  CONSTRAINT `fk_cuenta_ciudad1`
```



 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

```

FOREIGN KEY (`ciudad`)
REFERENCES `banco_bogota`.`ciudad` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

CREATE TABLE IF NOT EXISTS `banco_bogota`.`transaccion` (
  `id` INT(6) NOT NULL,
  `valor` DOUBLE ZEROFILL NOT NULL,
  `fecha` DATETIME NOT NULL,
  `tipo` INT(6) NOT NULL,
  `cuenta` INT(6) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_transaccion_tipo_transaccion_idx` (`tipo` ASC) ,
  INDEX `fk_transaccion_cuenta1_idx` (`cuenta` ASC) ,
  CONSTRAINT `fk_transaccion_tipo_transaccion`
    FOREIGN KEY (`tipo`)
      REFERENCES `banco_bogota`.`tipo_transaccion` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_transaccion_cuenta`
    FOREIGN KEY (`cuenta`)
      REFERENCES `banco_bogota`.`cuenta` (`num`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


```

#### *Banco Medellín DDL*

```

CREATE SCHEMA IF NOT EXISTS `banco_medellin` DEFAULT
CHARACTER SET utf8 ;

```


 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

```
CREATE TABLE IF NOT EXISTS `banco_medellin`.`ciudad` (
  `id` INT(6) NOT NULL,
  `nombre` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `banco_medellin`.`tipo_transaccion` (
  `id` INT(6) NOT NULL,
  `nombre` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `banco_medellin`.`cuenta` (
  `num` INT(6) NOT NULL,
  `cedula` VARCHAR(12) NOT NULL,
  `nombre` VARCHAR(45) NOT NULL,
  `telefono` VARCHAR(45) NOT NULL,
  `saldo` DOUBLE NOT NULL,
  `ciudad` INT(6) NOT NULL,
  PRIMARY KEY (`num`),
  INDEX `fk_cuenta_ciudad1_idx` (`ciudad` ASC) ,
  CONSTRAINT `fk_cuenta_ciudad1`
    FOREIGN KEY (`ciudad`)
    REFERENCES `banco_medellin`.`ciudad` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `banco_medellin`.`transaccion` (
```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

```

`id` INT(6) NOT NULL,
`valor` DOUBLE ZEROFILL NOT NULL,
`fecha` DATETIME NOT NULL,
`tipo` INT(6) NOT NULL,
`cuenta` INT(6) NOT NULL,
PRIMARY KEY (`id`),
INDEX `fk_transaccion_tipo_transaccion_idx` (`tipo` ASC),
INDEX `fk_transaccion_cuenta1_idx` (`cuenta` ASC),
CONSTRAINT `fk_transaccion_tipo_transaccion`
  FOREIGN KEY (`tipo`)
    REFERENCES `banco_medellin`.`tipo_transaccion` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
CONSTRAINT `fk_transaccion_cuenta`
  FOREIGN KEY (`cuenta`)
    REFERENCES `banco_medellin`.`cuenta` (`num`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

#### *Datacenter DDL*

```
CREATE SCHEMA IF NOT EXISTS `datacenter` DEFAULT CHARACTER SET utf8 ;
```

```

CREATE TABLE IF NOT EXISTS `datacenter`.`ciudad` (
  `id` INT(6) NOT NULL,
  `nombre` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))

```


```
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = utf8;
```

```

CREATE TABLE IF NOT EXISTS `datacenter`.`tipo_transaccion` (
  `id` INT(6) NOT NULL,
  `nombre` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))

```


 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

```
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

CREATE TABLE IF NOT EXISTS `datacenter`.`cuenta` (
  `num` INT(6) NOT NULL,
  `saldo` DOUBLE NOT NULL,
  PRIMARY KEY (`num`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

CREATE TABLE IF NOT EXISTS `datacenter`.`auditoria` (
  `id` INT(6) NOT NULL,
  `valor` DOUBLE ZEROFILL NOT NULL,
  `fecha` DATETIME NOT NULL,
  `tipo` INT(6) NOT NULL,
  `cuenta` INT(6) NOT NULL,
  `ciudad` INT(6) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_transaccion_tipo_transaccion_idx2` (`tipo` ASC),
  INDEX `fk_transaccion_cuenta1_idx2` (`cuenta` ASC),
  INDEX `fk_auditoria_ciudad1_idx2` (`ciudad` ASC),
  CONSTRAINT `fk_transaccion_tipo_transaccion2`
    FOREIGN KEY (`tipo`)
    REFERENCES `datacenter`.`tipo_transaccion` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_transaccion_cuenta2`
    FOREIGN KEY (`cuenta`)
    REFERENCES `datacenter`.`cuenta` (`num`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_auditoria_ciudad2`
    FOREIGN KEY (`ciudad`)
    REFERENCES `datacenter`.`ciudad` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

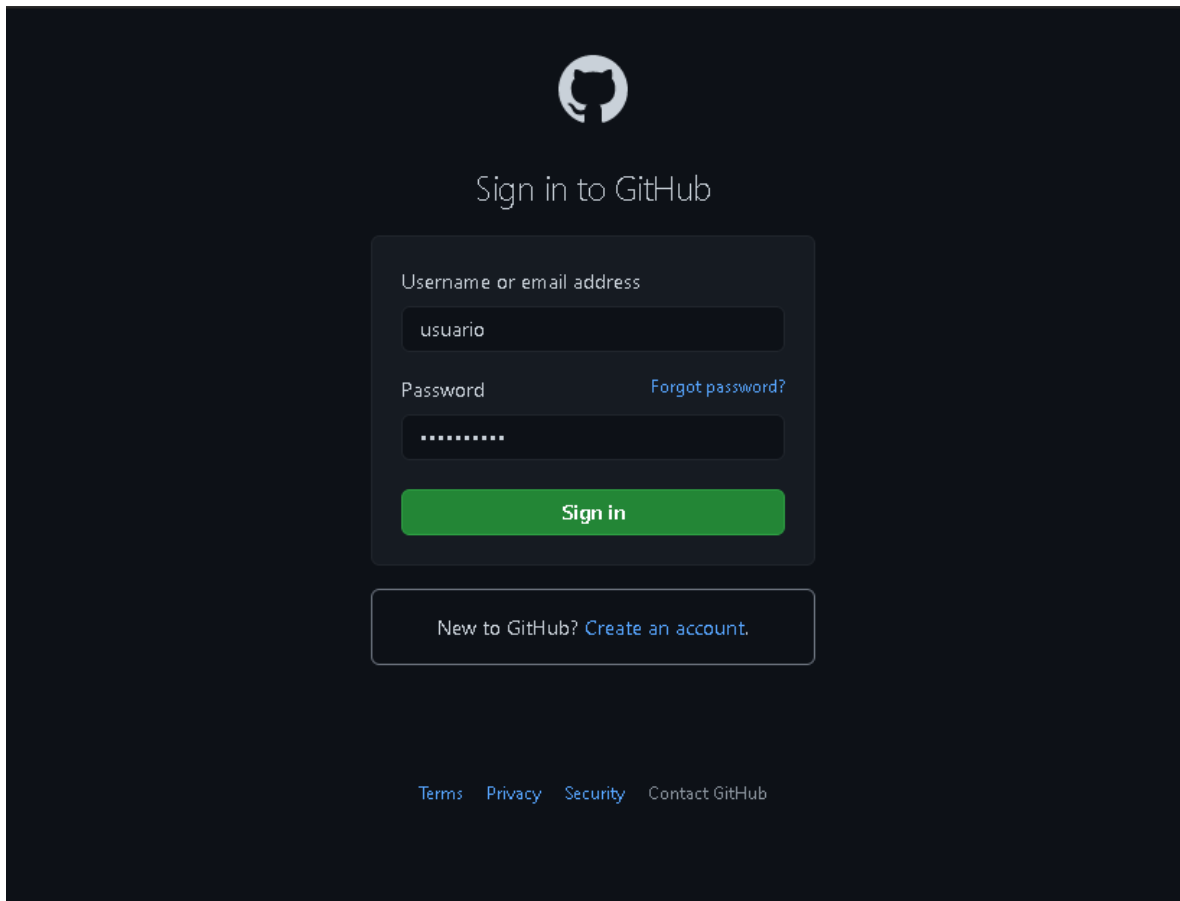
El usuario para la manipulación de las bases de datos será **Root** con la contraseña que previamente definimos en el proceso de instalación de MySQL.

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------


### Creación de repositorio

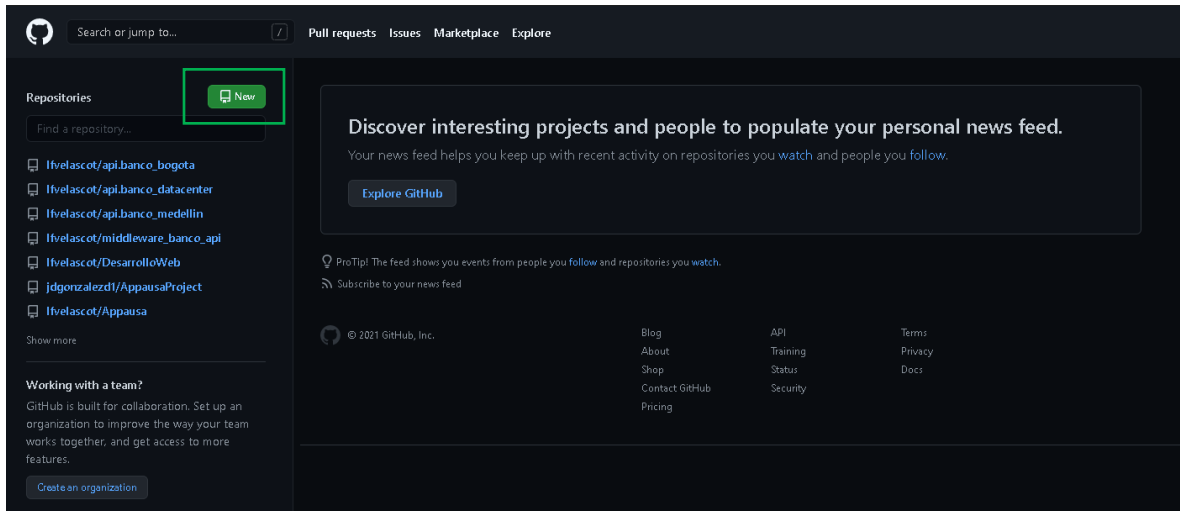
Antes de la creación de cada una de las APIs, nos enfocaremos en crear los repositorios en donde se almacenarán todos los archivos para cada API. Para esto, ingresaremos a nuestra cuenta de GitHub con el usuario que creamos previamente


<https://github.com/login>



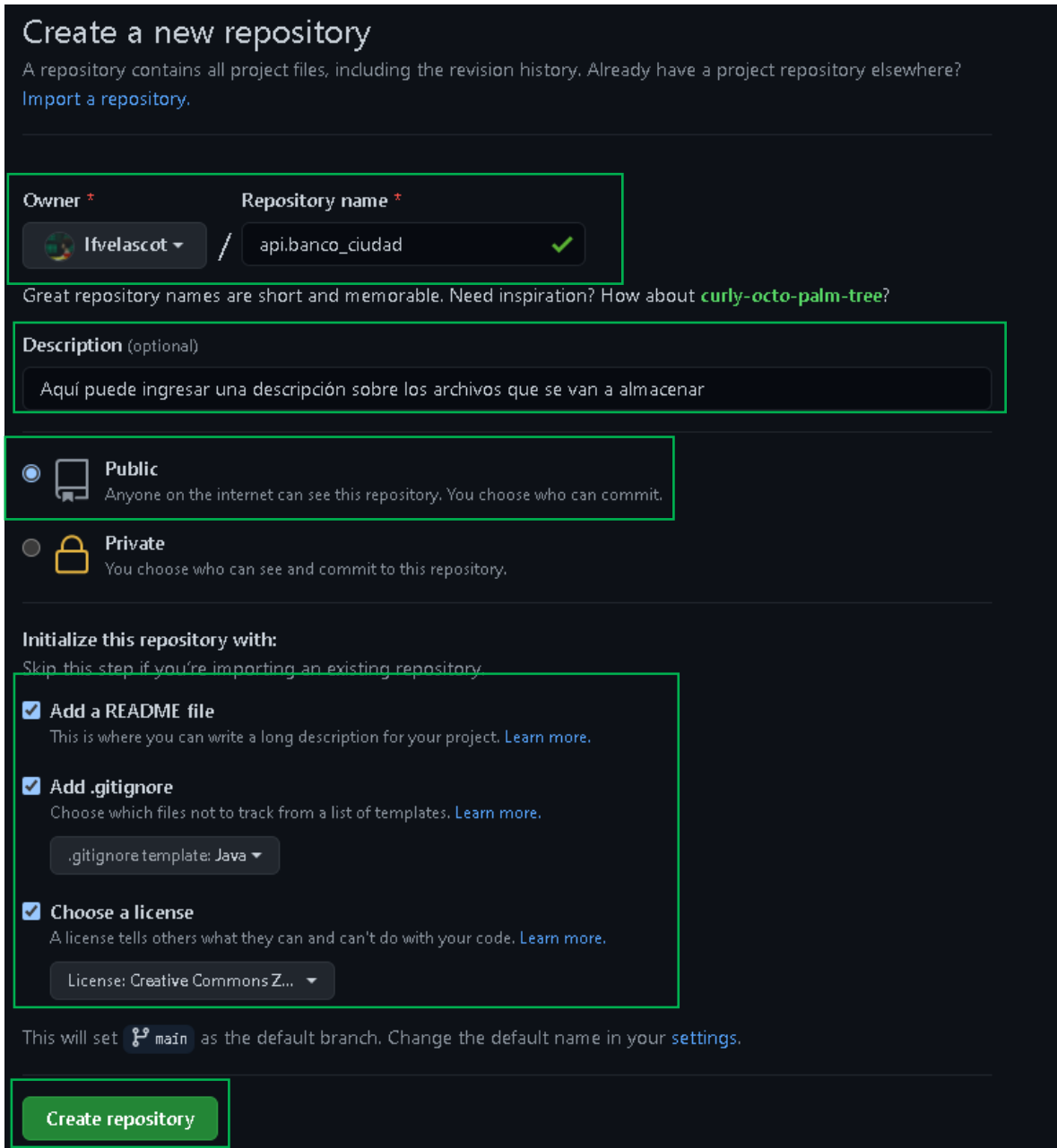
Ya dentro de nuestra cuenta, nos encontraremos en el Dashboard, para crear un nuevo repositorio daremos clic en el botón **New**:

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------



 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
--	---	--	----------


Dentro del apartado de crear un nuevo repositorio debemos definir los siguientes datos:



**Create a new repository**  
 A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

---

**Owner \*** **Repository name \***

 Ifvelascot / api.banco\_ciudad ✓


Great repository names are short and memorable. Need inspiration? How about [curly-octo-palm-tree?](#)


---

**Description (optional)**

Aquí puede ingresar una descripción sobre los archivos que se van a almacenar

---

☒  **Public**  
 Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
 You choose who can see and commit to this repository.

---


**Initialize this repository with:**  
[Skip this step if you're importing an existing repository](#)

☒ **Add a README file**  
 This is where you can write a long description for your project. [Learn more.](#)

☒ **Add .gitignore**  
 Choose which files not to track from a list of templates. [Learn more.](#)  
 .gitignore template: Java ▾

☒ **Choose a license**  
 A license tells others what they can and can't do with your code. [Learn more.](#)  
 License: Creative Commons Z... ▾


---

This will set  **main** as the default branch. Change the default name in your [settings](#).

---

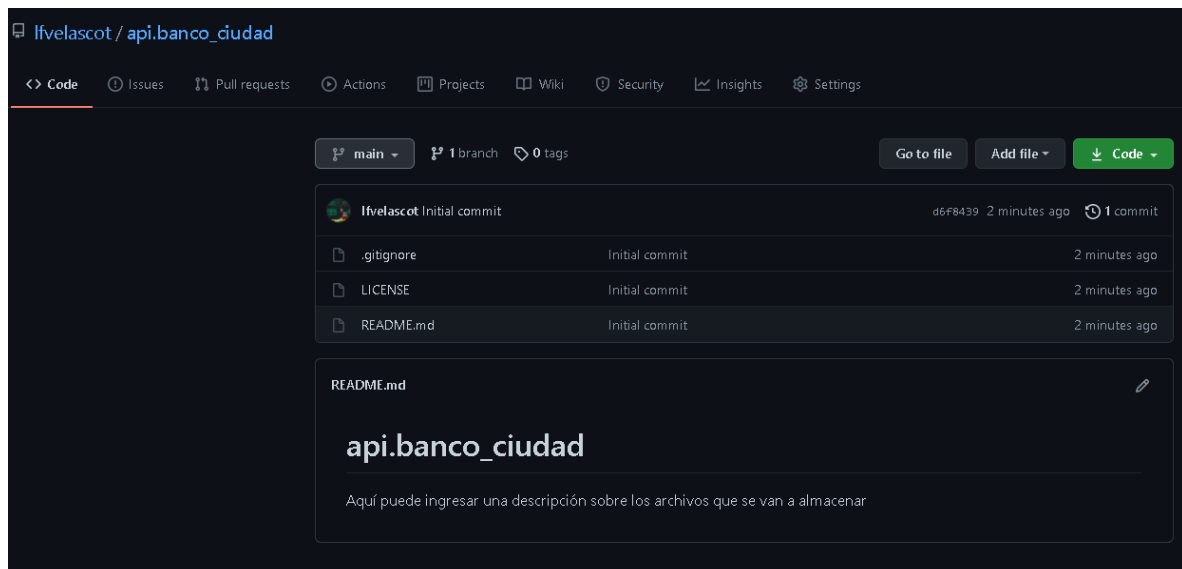
**Create repository**

- Repository name: aquí ingresaremos el nombre del API que se va a alojar en el repositorio. Este ejercicio lo vamos a realizar 4 veces para los siguientes componentes y requieren que cada repositorio se identifique de la siguiente forma:


 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

- API Datacenter = api.banco\_datacenter
- API Banco de Bogota = api.banco\_bogota
- API Banco de Medellin = api.banco\_medellin
- Middleware del banco = middleware\_banco\_api
- Description: Aquí puede dar una descripción general sobre los archivos que se almacenaran en el repositorio.
- El repositorio debe estar en Publico si queremos que cualquier persona tenga acceso a este y el código almacenado, de lo contrario, lo puede definir como privado para limitar quien puede tener acceso a este.
- Add .README file: seleccionaremos esta opción para que se cree un archivo .readme con la descripción del repositorio.
- Add .gitignore: seleccionaremos esta opción para que definamos que archivos queremos que se almacenen en nuestro repositorio, definiendo que el tipo de archivos sea Java para que no ignore los archivos.java del proyecto.
- Choose licence: este paso es opcional, pero en el ámbito del desarrollo profesional debemos tener en cuenta la existencia de licencias sobre la autoría y propiedad intelectual del código que creemos, en este caso definiremos como licencia Creative Commons Zero.

Al finalizar con todos los datos de configuración de nuestro repositorio daremos clic en **Create Repository**, siendo enviados a la página de nuestro repositorio, confirmando así que el proceso se realizó correctamente.



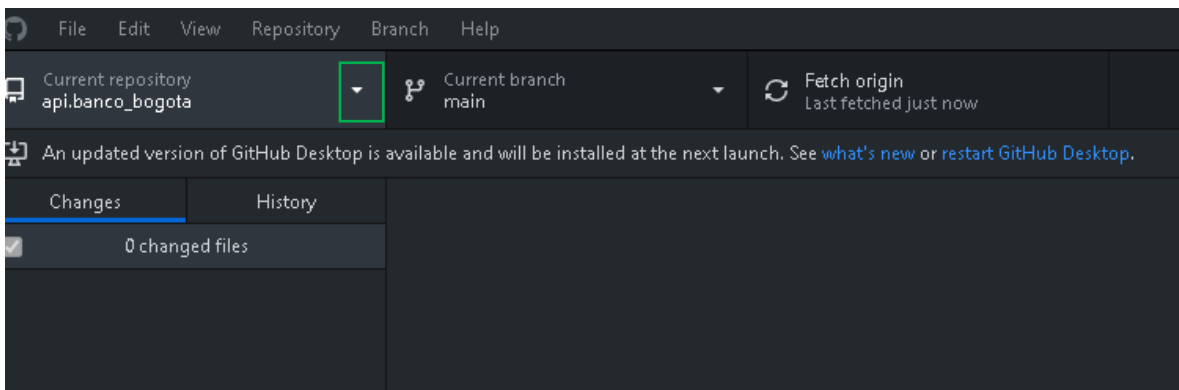


 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	---	----------

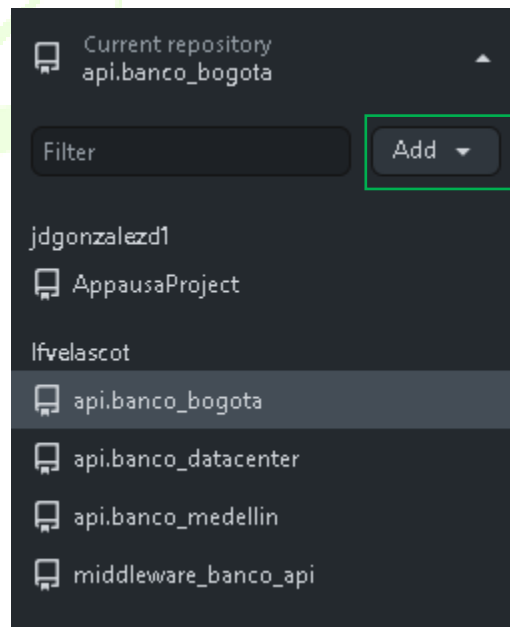
### *Clonación del repositorio*

Con el repositorio creado, vamos a abrir GitHub Desktop, y realizaremos la clonación del repositorio, esto con el fin de que podamos controlar desde nuestro escritorio las actualizaciones.


En la barra superior de Github Desktop nos encontraremos con las herramientas y un listado de los repositorios que tengamos existentes. Daremos click en la pestaña del listado de repositorios.

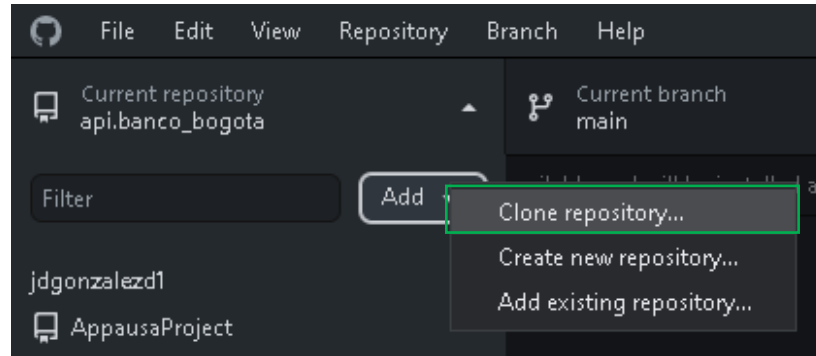


Dentro del listado de Repositorios, daremos clic en el botón Add:

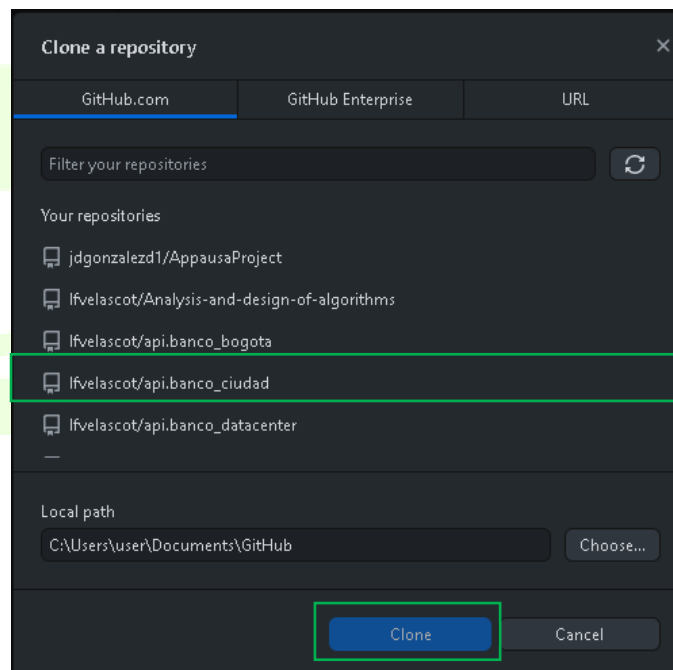


En el listado que nos aparece al presionar el boton, daremos click en la opcion Clone repository:


 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------

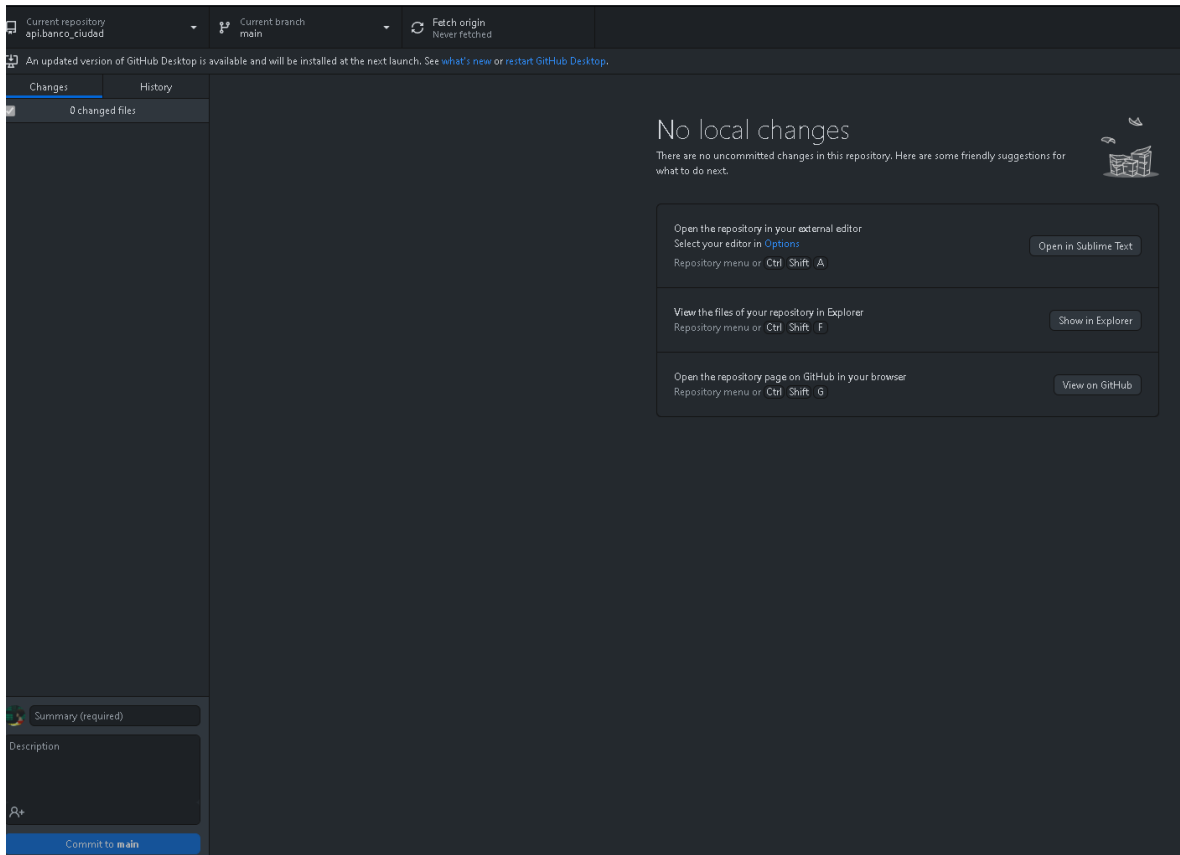


Se nos abrirá una ventana con las distintas fuentes de repositorios Git a las que tenemos acceso, en el primer apartado (Github.com) buscaremos el repositorio que creamos previamente, lo seleccionaremos y luego daremos clic en el botón **Clone**:



Al finalizar el proceso de clonación, se nos mostrara el dashboard del repositorio recién clonado:

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
--	---	---	----------




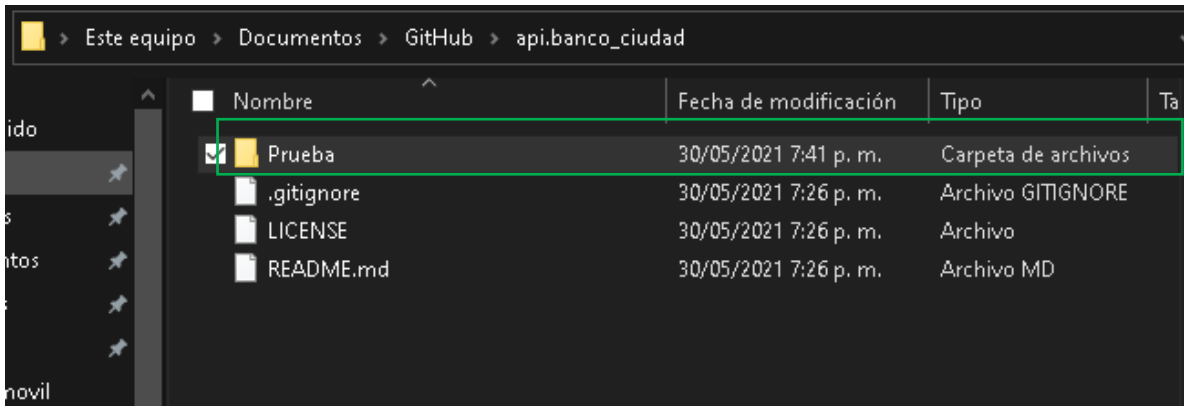
Nota: se debe repetir el proceso de clonación con cada uno de los repositorios creados.

### *Manipulación del repositorio*

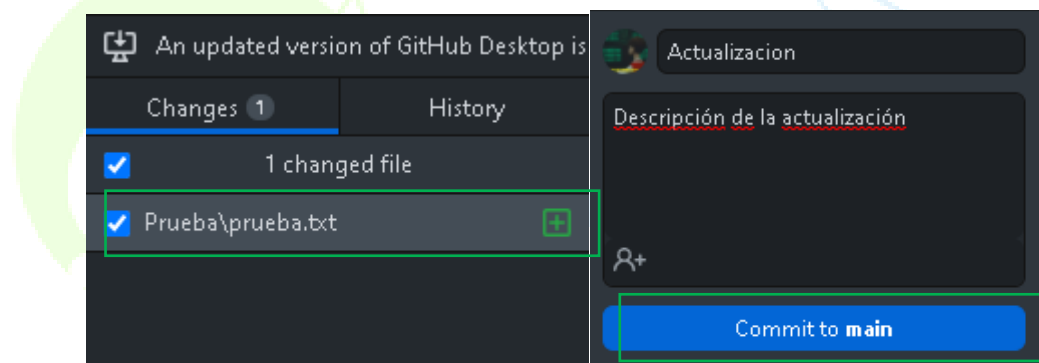
Podemos realizar las siguientes acciones sobre nuestro repositorio, de modo tal podamos controlar los archivos del repositorio:

1. Añadir archivos: nuestro repositorio se encuentra almacenado en la carpeta *documentos/github*, aquí veremos todos los repositorios que tenemos. Si solo queremos añadir un archivo, podemos crearlo directamente dentro de la carpeta de nuestro repositorio, añadiendo también carpetas.

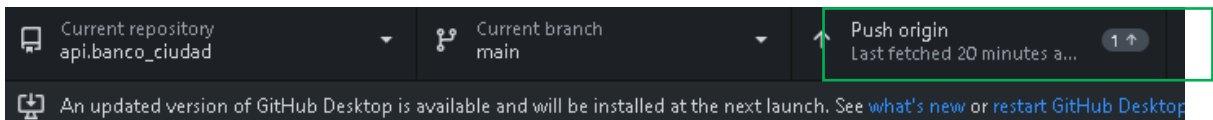
 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
--	---	--	----------



Cuando añadimos nuestros archivos, volvemos al repositorio en Github Desktop, veremos que en la tabla de **Changes** estará el archivo que añadimos, para actualizar el estado de nuestro repositorio, solo debemos añadir en la parte inferior una descripción de la actualización del repositorio daremos clic en **Commit to Main**.




Luego nos dirigiremos a la barra superior y encontraremos que este habilitado el botón **Push Origin**, (Origin hace referencia al repositorio alojado en Github).

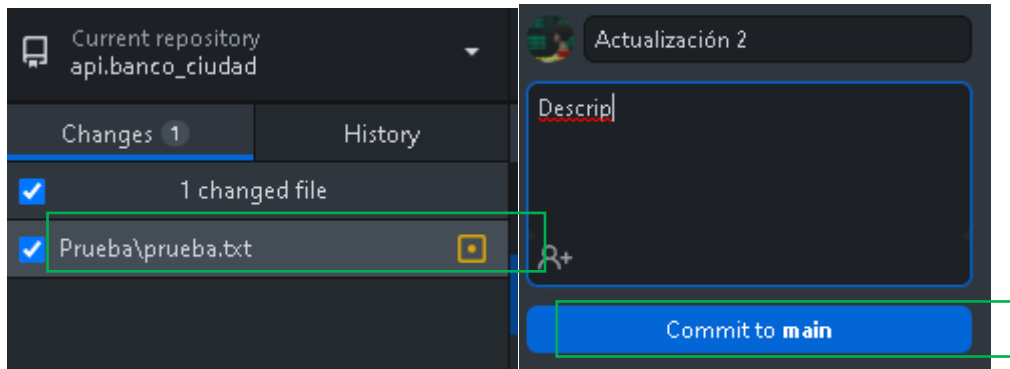


Daremos clic en el botón, lo cual actualizara todos los archivos en nuestro repositorio.

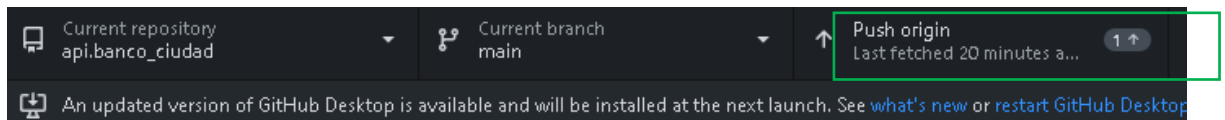
2. Actualizar archivos: Si realizamos cualquier modificación a los archivos alojados en la carpeta que hace referencia a nuestro repositorio, solo debemos dirigirnos a Github Desktop, en el listado de Changes veremos reflejadas las modificaciones realizadas, para que estas sean actualizadas

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

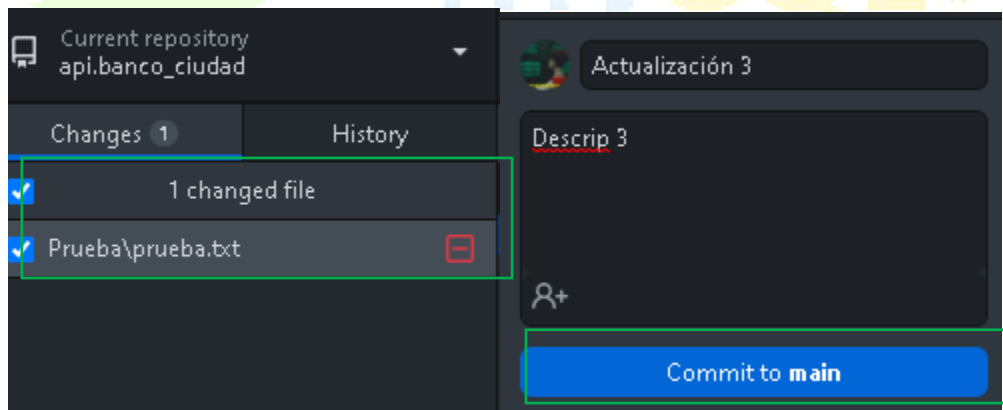
en nuestro repositorio, solo debemos llenar los datos del formulario de la parte inferior, y daremos clic en Commit to main



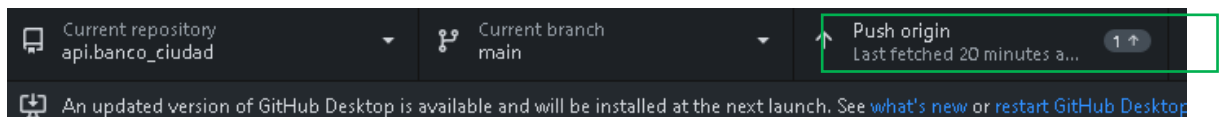
Finalmente actualizaremos nuestro repositorio en Github con los cambios realizados dando clic al botón en la barra superior de **Push Origin**.




3. Eliminar archivos: para eliminar archivos, solo debemos dirigirnos a la carpeta que hace referencia a nuestro repositorio, eliminaremos los archivos que requiramos, y volveremos a Github Desktop en donde encontraremos en la tabla de **Changes** que se elimino el archivo, para crear el commit a nuestro repositorio, nos dirigimos a la parte inferior y llenamos los datos del commit:



Con el commit creado, nos dirigiremos a la parte superior daremos click en el botón **Push Origin** para actualizar el repositorio.



 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

### Creación APIs


En el proceso de creación de APIs debemos tener en cuenta los esquemas físicos de cada una de las bases de datos creadas como resultado de la distribución, en base a estos crearemos las entidades requeridas para cada una de las APIs. Primero que todo, definiremos las características requeridas en la configuración de cada API:

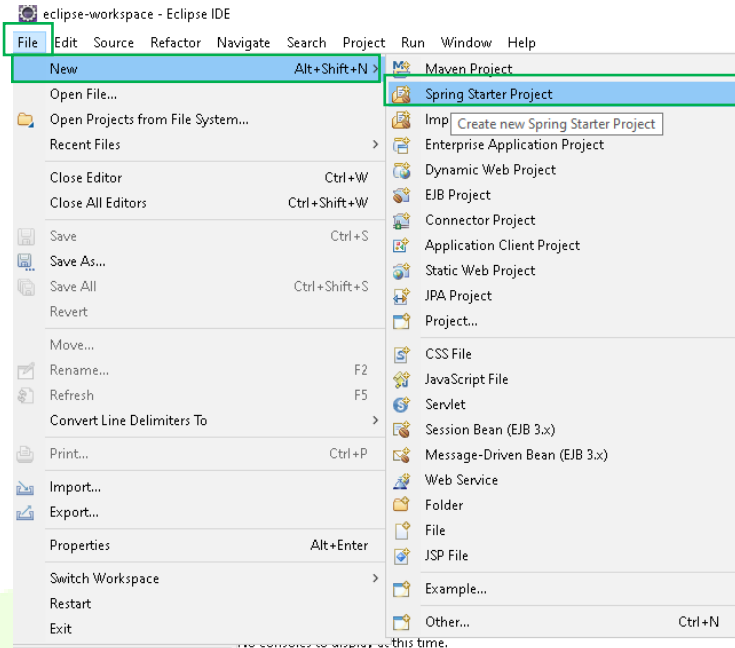
Tabla 1 Tabla de configuración

Nombre API	Dependencias	Puerto	Credenciales para la base de datos
api.banco_bogota	- MySQL Driver	3001	Usuario: root
api.banco_medellin	- REST Repositories - Spring Boot DevTools	3002	Contraseña: definida en el proceso de instalación
api.datacenter	- Spring Data JPA - Spring Web	3003	Se pueden crear usuarios para cada base de datos (por seguridad)
api.middleware	- REST Repositories - Spring Boot DevTools - Spring Data JPA - Spring Web	3004	No requeridos

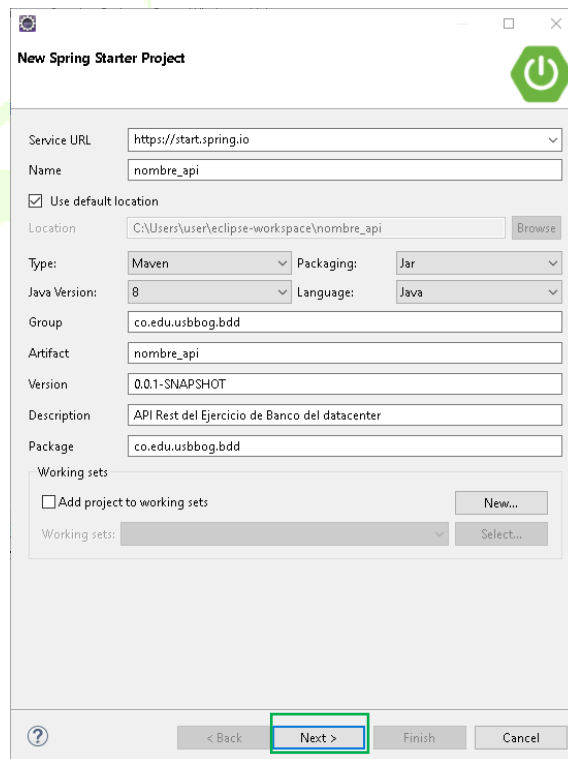
### Creación del proyecto

Este proceso será análogo para todos los proyectos de cada API, teniendo como variación los datos especificados en la anterior tabla. Para crear el proyecto en Eclipse, por favor tenga presente la previa instalación de Spring Tools Suit 4 (especificado en el apartado de instalación de software). Dentro de eclipse nos dirigiremos a la barra superior y daremos clic en *file / new / Spring Starter Project*.

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------



Dentro de la ventana, vamos a definir los siguientes datos:




The image shows the 'New Spring Starter Project' dialog box. The fields are filled out as follows:

- Service URL: <https://start.spring.io>
- Name: nombre\_api
- Use default location: ☒
- Location: C:\Users\user\workspace\nombre\_api
- Type: Maven
- Packaging: Jar
- Java Version: 8
- Language: Java
- Group: co.edu.usbbog.bdd
- Artifact: nombre\_api
- Version: 0.0.1-SNAPSHOT
- Description: API Rest del Ejercicio de Banco del datacenter
- Package: co.edu.usbbog.bdd
- Working sets: ☐ Add project to working sets
- Working sets:  Select...

The 'Next >' button is highlighted with a green box.

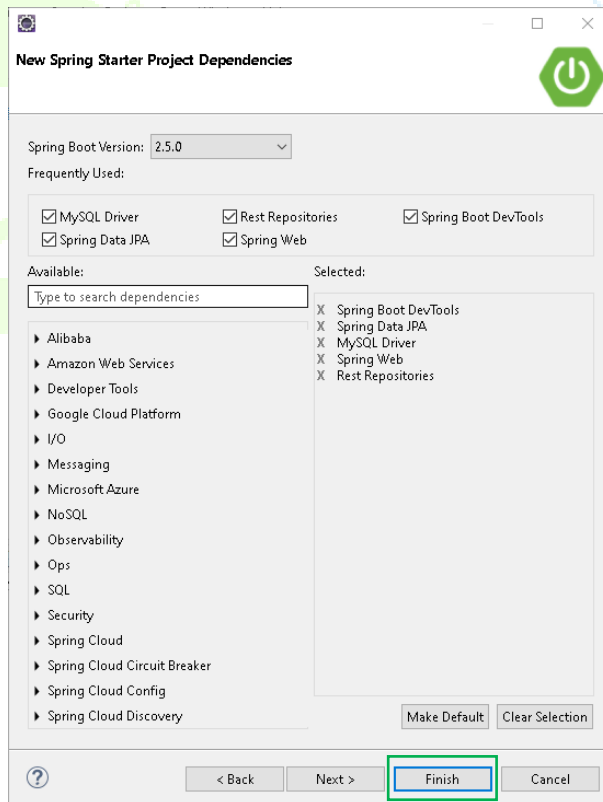
- Name: nombre de la API (ver tabla)

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------

- Type: Maven
- Packaging: Jar
- Java Version: 8
- Language: Java
- Group: co.edu.usbbog.bdd
- Package: co.edu.usbbog.bdd


Luego daremos clic en Next para añadir las dependencias del proyecto. En esta ventana vamos a buscar las siguientes dependencias:

- MySQL Driver
- REST Repositories
- Spring Boot DevTools
- Spring Data JPA
- Spring Web



Daremos clic en el botón **Finish** para terminar el proceso de creación del proyecto



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

### API Banco Bogotá

Ya contemplado el proceso de creación del proyecto, vamos a definir el proceso de configuración y desarrollo de los componentes y módulos requeridos.

#### Application.properties

En este archivo se definen todos los datos relacionados con el puerto por medio del cual estará disponible la API y las configuraciones requeridas para el uso de la base de datos:

```
server.port=3001 // Puerto de acceso
spring.jpa.database=MYSQL // Tipo de SGBD
spring.jpa.hibernate.ddl-auto=Update // Define si la modificación de las entitys
definirá una modificación en la base de datos
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver // nombre del
driver
spring.datasource.url =
jdbc:mysql://localhost:3306/banco_bogota?useUnicode=true&useJDBCCompliantTimezon
eShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC // URL o PATH de
acceso a la base de datos
spring.datasource.username=root // Usuario de la base de datos
spring.datasource.password=password // contraseña del usuario
spring.jpa.open-in-view=true // Visualización de sentencias SQL
```

#### Model

Este paquete se creará en el paquete *co.edu.usbbog.bdd* y será en el cual se definan todas las entidades que representaran cada una de las tablas, relaciones y claves primarias contenidas en la base de datos.

#### Ciudad


Esta entidad representara la tabla Ciudad contenida en la base de datos banco\_bogota, apoyándose en las anotaciones de persistencia para definir la estructura de los datos en la tabla:

```
package co.edu.usbbog.bdd.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Ciudad {

    @Id
    private long id;
    @Column
    private String nombre;
```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```

public Ciudad(long id, String nombre) {
    super();
    this.id = id;
    this.nombre = nombre;
}

public Ciudad() {
    super();
    this.id = 0;
    this.nombre = "";
}

public long getId() {
    return id;
}

public void setId(long id) {
    this.id = id;
}


public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + (int) (id ^ (id >>> 32));
    result = prime * result + ((nombre == null) ? 0 :
nombre.hashCode());
    return result;
}

@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    Ciudad other = (Ciudad) obj;

```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

```

        if (id != other.id)
            return false;
        if (nombre == null) {
            if (other.nombre != null)
                return false;
        } else if (!nombre.equals(other.nombre))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "Ciudad [id=" + id + ", nombre=" + nombre + "];"
    }
}

```

### Cuenta

Esta entidad representara la tabla Cuenta contenida en la base de datos banco\_bogota, apoyándose en las anotaciones de persistencia para definir la estructura de los datos en la tabla, aquí se debe tener en cuenta la asociación de la tabla Ciudad, en este caso, asociándolo a la clase/Entidad definida previamente:

```


package co.edu.usbbog.bdd.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

@Entity
public class Cuenta {

    @Id
    private long num;
    @Column
    private String cedula;
    @Column
    private String nombre;
    @Column
    private String telefono;
    @Column
    private Double saldo;
    @JoinColumn(name = "ciudad", referencedColumnName = "id", nullable =
false)
    @ManyToOne(optional = false)
    private Ciudad ciudad;
}

```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```

    public Cuenta(long num, String cedula, String nombre, String telefono,
Double saldo) {
        super();
        this.num = num;
        this.cedula = cedula;
        this.nombre = nombre;
        this.telefono = telefono;
        this.saldo = saldo;
    }

    public Cuenta() {
        super();
        this.num = 0;
        this.cedula = "";
        this.nombre = "";
        this.telefono = "";
        this.saldo = 0.0;
        this.ciudad = null;
    }

    public long getNum() {
        return num;
    }

    public void setNum(long num) {
        this.num = num;
    }

    public String getCedula() {
        return cedula;
    }

    public void setCedula(String cedula) {
        this.cedula = cedula;
    }


    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getTelefono() {
        return telefono;
    }

    public void setTelefono(String telefono) {

```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```

        this.telefono = telefono;
    }

    public Double getSaldo() {
        return saldo;
    }

    public void setSaldo(Double saldo) {
        this.saldo = saldo;
    }


    public Ciudad getCiudad() {
        return ciudad;
    }

    public void setCiudad(Ciudad ciudad) {
        this.ciudad = ciudad;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((cedula == null) ? 0 :
cedula.hashCode());
        result = prime * result + ((ciudad == null) ? 0 :
ciudad.hashCode());
        result = prime * result + ((nombre == null) ? 0 :
nombre.hashCode());
        result = prime * result + (int) (num ^ (num >>> 32));
        result = prime * result + ((saldo == null) ? 0 : saldo.hashCode());
        result = prime * result + ((telefono == null) ? 0 :
telefono.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Cuenta other = (Cuenta) obj;
        if (cedula == null) {
            if (other.cedula != null)
                return false;
        } else if (!cedula.equals(other.cedula))

```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

```

        return false;
    if (ciudad == null) {
        if (other.ciudad != null)
            return false;
    } else if (!ciudad.equals(other.ciudad))
        return false;
    if (nombre == null) {
        if (other.nombre != null)
            return false;
    } else if (!nombre.equals(other.nombre))
        return false;
    if (num != other.num)
        return false;
    if (saldo == null) {
        if (other.saldo != null)
            return false;
    } else if (!saldo.equals(other.saldo))
        return false;
    if (telefono == null) {
        if (other.telefono != null)
            return false;
    } else if (!telefono.equals(other.telefono))
        return false;
    return true;
}

@Override
public String toString() {
    return "Cuenta [num=" + num + ", cedula=" + cedula + ", nombre=" +
nombre + ", telefono=" + telefono
        + ", saldo=" + saldo + ", ciudad=" + ciudad.toString()
+ "];"
}
}

```

### Tipo\_Transaccion

Esta entidad representara la tabla tipo\_transaccion contenida en la base de datos banco\_bogota, apoyándose en las anotaciones de persistencia para definir la estructura de los datos en la tabla:


```

package co.edu.usbbog.bdd.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class TipoTransaccion {
    @Id

```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```

private long id;
private String nombre;

public TipoTransaccion(long id, String nombre) {
    super();
    this.id = id;
    this.nombre = nombre;
}

public TipoTransaccion() {
    super();
    this.id = 0;
    this.nombre = "";
}

public long getId() {
    return id;
}

public void setId(long id) {
    this.id = id;
}


public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + (int) (id ^ (id >>> 32));
    result = prime * result + ((nombre == null) ? 0 :
nombre.hashCode());
    return result;
}

@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;

```

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------

```

        TipoTransaccion other = (TipoTransaccion) obj;
        if (id != other.id)
            return false;
        if (nombre == null) {
            if (other.nombre != null)
                return false;
        } else if (!nombre.equals(other.nombre))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "TipoTransaccion [id=" + id + ", nombre=" + nombre + "];"
    }
}

```

### Transacción

Esta entidad representara la tabla Transaccion contenida en la base de datos banco\_bogota, apoyándose en las anotaciones de persistencia para definir la estructura de los datos en la tabla, teniendo en cuenta las relaciones con las tablas de Cuenta y Tipo\_transaccion, definidas como objetos relacionados a las clases/entidades:

```

package co.edu.usbbog.bdd.model;

import java.time.LocalDateTime;


import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

import com.fasterxml.jackson.annotation.JsonFormat;

@Entity
public class Transaccion {
    @Id
    private long id;
    @Column
    private double valor;
    @Column(nullable = false, name = "fecha", columnDefinition = "DATETIME")
    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss")
    private LocalDateTime fecha;
    @JoinColumn(name = "tipo", referencedColumnName = "id", nullable = false)
    @ManyToOne(optional = false)
    private TipoTransaccion tipo;
}

```



 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
--	---	---	----------

```


@JoinColumn(name = "cuenta", referencedColumnName = "num", nullable =
false)
@ManyToOne(optional = false)
private Cuenta cuenta;

public Transaccion(long id, double valor, LocalDateTime fecha) {
    super();
    this.id = id;
    this.valor = valor;
    this.fecha = fecha;
}

public Transaccion() {
    super();
    this.id = 0;
    this.valor = 0.0;
    this.fecha = null;
    this.tipo = null;
    this.cuenta = null;
}

public long getId() {
    return id;
}
public void setId(long id) {
    this.id = id;
}
public double getValor() {
    return valor;
}
public void setValor(double valor) {
    this.valor = valor;
}
public LocalDateTime getFecha() {
    return fecha;
}
public void setFecha(LocalDateTime fecha) {
    this.fecha = fecha;
}
public TipoTransaccion getTipo() {
    return tipo;
}
public void setTipo(TipoTransaccion tipo) {
    this.tipo = tipo;
}
public Cuenta getCuenta() {

```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```


        return cuenta;
    }

    public void setCuenta(Cuenta cuenta) {
        this.cuenta = cuenta;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((cuenta == null) ? 0 :
cuenta.hashCode());
        result = prime * result + ((fecha == null) ? 0 : fecha.hashCode());
        result = prime * result + (int) (id ^ (id >>> 32));
        result = prime * result + ((tipo == null) ? 0 : tipo.hashCode());
        long temp;
        temp = Double.doubleToLongBits(valor);
        result = prime * result + (int) (temp ^ (temp >>> 32));
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Transaccion other = (Transaccion) obj;
        if (cuenta == null) {
            if (other.cuenta != null)
                return false;
        } else if (!cuenta.equals(other.cuenta))
            return false;
        if (fecha == null) {
            if (other.fecha != null)
                return false;
        } else if (!fecha.equals(other.fecha))
            return false;
        if (id != other.id)
            return false;
        if (tipo == null) {
            if (other.tipo != null)
                return false;

```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

```

        } else if (!tipo.equals(other.tipo))
            return false;
        if (Double.doubleToLongBits(valor) !=
Double.doubleToLongBits(other.valor))
            return false;
        return true;
    }

    @Override
    public String toString() {
        return "Transaccion [id=" + id + ", valor=" + valor + ", fecha=" +
fecha + ", tipo=" + tipo.toString() + ", cuenta="
+ cuenta.toString() + "];"
    }
}

```

### Repository

Este paquete se creará en el paquete *co.edu.usbbog.bdd* y será en el cual se definan las interfaces por medio de las cuales tendremos acceso a los métodos definidos en la clase *JpaRepository*, esto lo realizaremos con cada una de las entidades.

### ICiudad

Clase de interfaz para la entidad Ciudad que extenderá de *JpaRepository*:

```

package co.edu.usbbog.bdd.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import co.edu.usbbog.bdd.model.Ciudad;

public interface ICiudad extends JpaRepository<Ciudad, Long> {
}

```

### ICuenta

Clase de interfaz para la entidad Cuenta que extenderá de *JpaRepository*:


```

package co.edu.usbbog.bdd.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import co.edu.usbbog.bdd.model.Cuenta;

public interface ICuenta extends JpaRepository<Cuenta, Long> {
}

```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

### ITipoTransaccion

Clase de interfaz para la entidad TipoTransaccion que extenderá de JpaRepository:

```
package co.edu.usbbog.bdd.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import co.edu.usbbog.bdd.model.TipoTransaccion;

public interface ITipoTransaccion extends JpaRepository<TipoTransaccion, Long>{
}
```

### ITransaccion

Clase de interfaz para la entidad Transaccion que extenderá de JpaRepository:

```
package co.edu.usbbog.bdd.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import co.edu.usbbog.bdd.model.Transaccion;

public interface ITransaccion extends JpaRepository<Transaccion, Long>{
}
```

### Rest


Este paquete se creará en el paquete *co.edu.usbbog.bdd* y será en el cual se definan los métodos a los cuales se podrán tener acceso por medio del mapeo de rutas y el uso de las interfaces para cada uno de los objetos previamente definidas en el paquete repository.

### CiudadController

Clase definida con la anotación **RestController** que contiene todos los métodos a los cuales podrán tener acceso mediante la ruta **localhost:3001/ciudad/**, ruta en la cual se podrán realizar las acciones creación, listado, búsqueda por ID, conteo, eliminación y actualización de los registros de ciudades contenidos en la tabla ciudad de la base de datos banco\_bogota, todo esto por medio de la interfaz ICiudad definida para la entidad Ciudad.

```
package co.edu.usbbog.bdd.rest;

import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import co.edu.usbbog.bdd.model.Ciudad;
import co.edu.usbbog.bdd.repository.ICiudad;

@RestController
@RequestMapping("/ciudad")
public class CiudadController {

    @Autowired
    private ICiudad ic;


    @PostMapping("/create")
    public void insertCiudad(@RequestBody Ciudad c) {
        ic.save(c);
    }

    @GetMapping("/all")
    public List<Ciudad> findAllCiudades() {
        List<Ciudad> l = ic.findAll();
        if (l.isEmpty() || l.equals(null)) {
            throw new RuntimeException("No hay ciudades registradas");
        } else {
            return l;
        }
    }

    @GetMapping("/find/{id}")
    public Optional<Ciudad> findCiudad(@PathVariable("id") long id) {
        Optional<Ciudad> ci = ic.findById(id);
        if (!ci.equals(null)) {
            return ci;
        } else {
            throw new RuntimeException("Ciudad identificada con el ID: "
+ id + " no encontrado");
        }
    }

    @GetMapping("/count")
    public long countCiudades() {
        long c = ic.count();
        if (c != 0) {
            return c;
        } else {
            throw new RuntimeException("No hay ciudades registradas");
        }
    }
}

```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

```

    }

    @DeleteMapping("/delete/{id}")
    public void deleteCiudad(@PathVariable("id") long id) {
        Optional<Ciudad> c = ic.findById(id);
        if (c.equals(null)) {
            throw new RuntimeException("Ciudad identificada con el ID: "
+ id + " no encontrado");
        } else {
            ic.deleteById(id);
        }
    }

    @PutMapping("/update")
    public void updateCiudad(@RequestBody Ciudad c) {
        Optional<Ciudad> ci = ic.findById(c.getId());
        if (ci.equals(null)) {
            throw new RuntimeException("Ciudad identificada con el ID: "
+ c.getId() + " no encontrado");
        } else {
            ic.save(c);
        }
    }
}

```

#### CuentaController


Clase definida con la anotación **RestController** que contiene todos los métodos a los cuales podrán tener acceso mediante la ruta **localhost:3001/cuenta/**, ruta en la cual se podrán realizar las acciones creación, listado, búsqueda por ID, conteo, eliminación y actualización de los registros de cuentas contenidas en la tabla cuenta de la base de datos banco\_bogota, todo esto por medio de la interfaz ICuenta definida para la entidad Cuenta.

```

package co.edu.usbbog.bdd.rest;

import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import co.edu.usbbog.bdd.model.Cuenta;
import co.edu.usbbog.bdd.repository.ICuenta;

```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```

@RestController
@RequestMapping("/cuenta")
public class CuentaController {
    @Autowired
    private ICuenta ic;


    @PostMapping("/create")
    public void insertCuenta(@RequestBody Cuenta c) {
        ic.save(c);
    }

    @GetMapping("/all")
    public List<Cuenta> findAllCuentas() {
        List<Cuenta> l = ic.findAll();
        if (l.isEmpty() || l.equals(null)) {
            throw new RuntimeException("No hay cuentas registradas");
        } else {
            return l;
        }
    }

    @GetMapping("/find/{id}")
    public Optional<Cuenta> findCuenta(@PathVariable("id") long id) {
        Optional<Cuenta> cu = ic.findById(id);
        if (!cu.equals(null)) {
            return cu;
        } else {
            throw new RuntimeException("Cuenta identificada con el ID: "
+ id + " no encontrado");
        }
    }

    @GetMapping("/count")
    public long countCuenta() {
        long c = ic.count();
        if (c != 0) {
            return c;
        } else {
            throw new RuntimeException("No hay cuentas registradas");
        }
    }

    @DeleteMapping("/delete/{id}")
    public void deleteCuenta(@PathVariable("id") long id) {
        Optional<Cuenta> cu = ic.findById(id);
        if (!cu.equals(null)) {
            ic.deleteById(id);
        } else {
    
```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	---	----------

```

        throw new RuntimeException("Cuenta identificada con el ID: "
+ id + " no encontrado");
    }
}

@PutMapping("/update")
public void updateCuenta(@RequestBody Cuenta c) {
    Optional<Cuenta> cu = ic.findById(c.getNum());
    if (!cu.equals(null)) {
        ic.save(c);
    } else {
        throw new RuntimeException("Cuenta identificada con el ID: "
+ c.getNum() + " no encontrado");
    }
}
}

```

#### TipoTransaccionController

Clase definida con la anotación **RestController** que contiene todos los métodos a los cuales podrán tener acceso mediante la ruta **localhost:3001/tipotransaccion/**, ruta en la cual se podrán realizar las acciones creación, listado, búsqueda por ID, conteo, eliminación y actualización de los registros de tipos de transacciones contenidos en la tabla tipotransaccion de la base de datos banco\_bogota, todo esto por medio de la interfaz ITipoTransaccion definida para la entidad TipoTransaccion.

```

package co.edu.usbbog.bdd.rest;


import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import co.edu.usbbog.bdd.model.TipoTransaccion;
import co.edu.usbbog.bdd.repository.ITipoTransaccion;

@RestController
@RequestMapping("/tipotransaccion")
public class TipoTransaccionController {
    @Autowired
    private ITipoTransaccion itt;

    @PostMapping("/create")
    public void insertTipoTransaccion(@RequestBody TipoTransaccion tt) {

```



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```

        itt.save(tt);
    }

    @GetMapping("/all")
    public List<TipoTransaccion> findAllTipoTransacciones() {
        List<TipoTransaccion> l = itt.findAll();
        if (l.isEmpty() || l.equals(null)) {
            throw new RuntimeException("No hay tipos de transacciones
registradas");
        } else {
            return l;
        }
    }


    @GetMapping("/find/{id}")
    public Optional<TipoTransaccion> findTipoTransaccion(@PathVariable("id")
long id) {
        Optional<TipoTransaccion> tt = itt.findById(id);
        if (!tt.equals(null)) {
            return tt;
        } else {
            throw new RuntimeException("Tipo de Tramsaccion identificada
con el ID: " + id + " no encontrado");
        }
    }

    @GetMapping("/count")
    public long countTipoTransacciones() {
        long c = itt.count();
        if (c != 0) {
            return c;
        } else {
            throw new RuntimeException("No hay tipos de transacciones
registradas");
        }
    }

    @DeleteMapping("/delete/{id}")
    public void deleteTipoTransaccion(@PathVariable("id") long id) {
        Optional<TipoTransaccion> tt = itt.findById(id);
        if (!tt.equals(null)) {
            itt.deleteById(id);
        } else {
            throw new RuntimeException("Tipo de Tramsaccion identificada
con el ID: " + id + " no encontrado");
        }
    }

    @PutMapping("/update")

```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	---	----------

```

    public void updateTipoTransaccion(@RequestBody TipoTransaccion tt) {
        Optional<TipoTransaccion> t = itt.findById(tt.getId());
        if (!t.equals(null)) {
            itt.save(tt);
        } else {
            throw new RuntimeException("Tipo de Tramsaccion identificada con el ID: " + tt.getId() + " no encontrado");
        }
    }
}

```

### TransaccionController

Clase definida con la anotación **RestController** que contiene todos los métodos a los cuales podrán tener acceso mediante la ruta **localhost:3001/transaccion/**, ruta en la cual se podrán realizar las acciones creación, listado, búsqueda por ID, conteo, eliminación y actualización de los registros de transacciones contenidos en la tabla transaccion de la base de datos banco\_bogota, todo esto por medio de la interfaz ITransaccion definida para la entidad Transaccion.

```

package co.edu.usbbog.bdd.rest;

import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import co.edu.usbbog.bdd.model.Transaccion;
import co.edu.usbbog.bdd.repository.ITransaccion;


@RestController
@RequestMapping("/transaccion")
public class TransaccionController {

    @Autowired
    ITransaccion it;

    @PostMapping("/create")
    public void insertTransaccion(@RequestBody Transaccion c) {
        it.save(c);
    }

    @GetMapping("/all")
    public List<Transaccion> findAllTransacciones() {

```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```

        List<Transaccion> l = it.findAll();
        if (l.isEmpty() || l.equals(null)) {
            throw new RuntimeException("No hay transacciones
registradas");
        } else {
            return l;
        }
    }


    @GetMapping("/find/{id}")
    public Optional<Transaccion> findTransaccion(@PathVariable("id") long id)
{
        Optional<Transaccion> t = it.findById(id);
        if (!t.equals(null)) {
            return t;
        } else {
            throw new RuntimeException("Transaccion identificada con el
ID: " + id + " no encontrado");
        }
    }

    @GetMapping("/count")
    public long countTransaccion() {
        long c = it.count();
        if (c != 0) {
            return c;
        } else {
            throw new RuntimeException("No hay transacciones
registradas");
        }
    }

    @DeleteMapping("/delete/{id}")
    public void deleteTransaccion(@PathVariable("id") long id) {
        Optional<Transaccion> t = it.findById(id);
        if (!t.equals(null)) {
            it.deleteById(id);
        } else {
            throw new RuntimeException("Transaccion identificada con el
ID: " + id + " no encontrado");
        }
    }

    @PutMapping("/update")
    public void updateTransaccion(@RequestBody Transaccion c) {
        Optional<Transaccion> t = it.findById(c.getId());
        if (!t.equals(null)) {
            it.save(c);
        } else {

```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	---	----------

```

        throw new RuntimeException("Tramsaccion identificada con el
ID: " + c.getId() + " no encontrado");
    }
}
}

```

### API Banco Medellín

Ya contemplado el proceso de creación del proyecto, vamos a definir el proceso de configuración y desarrollo de los componentes y módulos requeridos.

### Application.properties

En este archivo se definen todos los datos relacionados con el puerto por medio del cual estará disponible la API y las configuraciones requeridas para el uso de la base de datos:

```

server.port=3002 // Puerto de acceso
spring.jpa.database=MYSQL // Tipo de SGBD
spring.jpa.hibernate.ddl-auto=Update // Define si la modificación de las entitys
definirá una modificación en la base de datos
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver // nombre del
driver
spring.datasource.url =
jdbc:mysql://localhost:3306/banco_medellin?useUnicode=true&useJDBCCompliantTimez
oneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC // URL o PATH de
acceso a la base de datos
spring.datasource.username=root // Usuario de la base de datos
spring.datasource.password=password // contraseña del usuario
spring.jpa.open-in-view=true // Visualización de sentencias SQL

```

### Model

### Repository


### Rest

Este paquete se creará en el paquete *co.edu.usbbog.bdd* y será en el cual se definan los métodos a los cuales se podrán tener acceso por medio del mapeo de rutas y el uso de las interfaces para cada uno de los objetos previamente definidas en el paquete repository.

### CiudadController

Clase definida con la anotación **RestController** que contiene todos los métodos a los cuales podrán tener acceso mediante la ruta **localhost:3002/ciudad/**, ruta en la cual se podrán realizar las acciones creación, listado, búsqueda por ID, conteo, eliminación y actualización de los registros de ciudades contenidos en la tabla ciudad de la base de datos banco\_ medellin, todo esto por medio de la interfaz ICiudad definida para la entidad Ciudad.

```
package co.edu.usbbog.bdd.rest;
```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```

import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import co.edu.usbbog.bdd.model.Ciudad;
import co.edu.usbbog.bdd.repository.ICiudad;

@RestController
@RequestMapping("/ciudad")
public class CiudadController {

    @Autowired
    private ICiudad ic;


    @PostMapping("/create")
    public void insertCiudad(@RequestBody Ciudad c) {
        ic.save(c);
    }

    @GetMapping("/all")
    public List<Ciudad> findAllCiudades() {
        List<Ciudad> l = ic.findAll();
        if (l.isEmpty() || l.equals(null)) {
            throw new RuntimeException("No hay ciudades registradas");
        } else {
            return l;
        }
    }

    @GetMapping("/find/{id}")
    public Optional<Ciudad> findCiudad(@PathVariable("id") long id) {
        Optional<Ciudad> ci = ic.findById(id);
        if (!ci.equals(null)) {
            return ci;
        } else {
            throw new RuntimeException("Ciudad identificada con el ID: "
+ id + " no encontrado");
        }
    }

    @GetMapping("/count")

```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

```

    public long countCiudades() {
        long c = ic.count();
        if (c != 0) {
            return c;
        } else {
            throw new RuntimeException("No hay ciudades registradas");
        }
    }

    @DeleteMapping("/delete/{id}")
    public void deleteCiudad(@PathVariable("id") long id) {
        Optional<Ciudad> c = ic.findById(id);
        if (c.equals(null)) {
            throw new RuntimeException("Ciudad identificada con el ID: "
+ id + " no encontrado");
        } else {
            ic.deleteById(id);
        }
    }

    @PutMapping("/update")
    public void updateCiudad(@RequestBody Ciudad c) {
        Optional<Ciudad> ci = ic.findById(c.getId());
        if (ci.equals(null)) {
            throw new RuntimeException("Ciudad identificada con el ID: "
+ c.getId() + " no encontrado");
        } else {
            ic.save(c);
        }
    }
}

```

#### CuentaController


Clase definida con la anotación **RestController** que contiene todos los métodos a los cuales podrán tener acceso mediante la ruta **localhost:3002/cuenta/**, ruta en la cual se podrán realizar las acciones creación, listado, búsqueda por ID, conteo, eliminación y actualización de los registros de cuentas contenidas en la tabla cuenta de la base de datos banco\_ medellin, todo esto por medio de la interfaz ICuenta definida para la entidad Cuenta.

```

package co.edu.usbbog.bdd.rest;

import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import co.edu.usbbog.bdd.model.Cuenta;
import co.edu.usbbog.bdd.repository.ICuenta;


@RestController
@RequestMapping("/cuenta")
public class CuentaController {
    @Autowired
    private ICuenta ic;

    @PostMapping("/create")
    public void insertCuenta(@RequestBody Cuenta c) {
        ic.save(c);
    }

    @GetMapping("/all")
    public List<Cuenta> findAllCuentas() {
        List<Cuenta> l = ic.findAll();
        if (l.isEmpty() || l.equals(null)) {
            throw new RuntimeException("No hay cuentas registradas");
        } else {
            return l;
        }
    }

    @GetMapping("/find/{id}")
    public Optional<Cuenta> findCuenta(@PathVariable("id") long id) {
        Optional<Cuenta> cu = ic.findById(id);
        if (!cu.equals(null)) {
            return cu;
        } else {
            throw new RuntimeException("Cuenta identificada con el ID: "
+ id + " no encontrado");
        }
    }

    @GetMapping("/count")
    public long countCuenta() {
        long c = ic.count();
        if (c != 0) {
            return c;
        } else {
            throw new RuntimeException("No hay cuentas registradas");
        }
    }
}
```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	---	----------

```

@DeleteMapping("/delete/{id}")
public void deleteCuenta(@PathVariable("id") long id) {
    Optional<Cuenta> cu = ic.findById(id);
    if (!cu.equals(null)) {
        ic.deleteById(id);
    } else {
        throw new RuntimeException("Cuenta identificada con el ID: "
+ id + " no encontrado");
    }
}

@PutMapping("/update")
public void updateCuenta(@RequestBody Cuenta c) {
    Optional<Cuenta> cu = ic.findById(c.getNum());
    if (!cu.equals(null)) {
        ic.save(c);
    } else {
        throw new RuntimeException("Cuenta identificada con el ID: "
+ c.getNum() + " no encontrado");
    }
}
}

```

#### TipoTransaccionController

Clase definida con la anotación **RestController** que contiene todos los métodos a los cuales podrán tener acceso mediante la ruta **localhost:3002/tipotransaccion/**, ruta en la cual se podrán realizar las acciones creación, listado, búsqueda por ID, conteo, eliminación y actualización de los registros de tipos de transacciones contenidos en la tabla tipotransaccion de la base de datos banco\_ medellin, todo esto por medio de la interfaz ITipoTransaccion definida para la entidad TipoTransaccion.


```

package co.edu.usbbog.bdd.rest;

import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import co.edu.usbbog.bdd.model.TipoTransaccion;
import co.edu.usbbog.bdd.repository.ITipoTransaccion;

```



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```

@RestController
@RequestMapping("/tipotransaccion")
public class TipoTransaccionController {
    @Autowired
    private ITipoTransaccion itt;

    @PostMapping("/create")
    public void insertTipoTransaccion(@RequestBody TipoTransaccion tt) {
        itt.save(tt);
    }


    @GetMapping("/all")
    public List<TipoTransaccion> findAllTipoTransacciones() {
        List<TipoTransaccion> l = itt.findAll();
        if (l.isEmpty() || l.equals(null)) {
            throw new RuntimeException("No hay tipos de transacciones
registradas");
        } else {
            return l;
        }
    }

    @GetMapping("/find/{id}")
    public Optional<TipoTransaccion> findTipoTransaccion(@PathVariable("id")
long id) {
        Optional<TipoTransaccion> tt = itt.findById(id);
        if (!tt.equals(null)) {
            return tt;
        } else {
            throw new RuntimeException("Tipo de Tramsaccion identificada
con el ID: " + id + " no encontrado");
        }
    }

    @GetMapping("/count")
    public long coundTipoTransacciones() {
        long c = itt.count();
        if (c != 0) {
            return c;
        } else {
            throw new RuntimeException("No hay tipos de transacciones
registradas");
        }
    }

    @DeleteMapping("/delete/{id}")
    public void deleteTipoTransaccion(@PathVariable("id") long id) {
        Optional<TipoTransaccion> tt = itt.findById(id);
        if (!tt.equals(null)) {

```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	---	----------

```

        itt.deleteById(id);
    } else {
        throw new RuntimeException("Tipo de Tramsaccion identificada
con el ID: " + id + " no encontrado");
    }
}

@PutMapping("/update")
public void updateTipoTransaccion(@RequestBody TipoTransaccion tt) {
    Optional<TipoTransaccion> t = itt.findById(tt.getId());
    if (!t.equals(null)) {
        itt.save(tt);
    } else {
        throw new RuntimeException("Tipo de Tramsaccion identificada
con el ID: " + tt.getId() + " no encontrado");
    }
}
}

```

#### TransaccionController

Clase definida con la anotación **RestController** que contiene todos los métodos a los cuales podrán tener acceso mediante la ruta **localhost:3002/transaccion/**, ruta en la cual se podrán realizar las acciones creación, listado, búsqueda por ID, conteo, eliminación y actualización de los registros de transacciones contenidos en la tabla transaccion de la base de datos banco\_medellin, todo esto por medio de la interfaz ITransaccion definida para la entidad Transaccion.

```


package co.edu.usbbog.bdd.rest;

import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import co.edu.usbbog.bdd.model.Transaccion;
import co.edu.usbbog.bdd.repository.ITransaccion;

@RestController
@RequestMapping("/transaccion")
public class TransaccionController {

    @Autowired
    ITransaccion it;
}

```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```

@PostMapping("/create")
public void insertTransaccion(@RequestBody Transaccion c) {
    it.save(c);
}


@GetMapping("/all")
public List<Transaccion> findAllTransacciones() {
    List<Transaccion> l = it.findAll();
    if (l.isEmpty() || l.equals(null)) {
        throw new RuntimeException("No hay transacciones
registradas");
    } else {
        return l;
    }
}

@GetMapping("/find/{id}")
public Optional<Transaccion> findTransaccion(@PathVariable("id") long id)
{
    Optional<Transaccion> t = it.findById(id);
    if (!t.equals(null)) {
        return t;
    } else {
        throw new RuntimeException("Transaccion identificada con el
ID: " + id + " no encontrado");
    }
}

@GetMapping("/count")
public long countTransaccion() {
    long c = it.count();
    if (c != 0) {
        return c;
    } else {
        throw new RuntimeException("No hay transacciones
registradas");
    }
}

@DeleteMapping("/delete/{id}")
public void deleteTransaccion(@PathVariable("id") long id) {
    Optional<Transaccion> t = it.findById(id);
    if (!t.equals(null)) {
        it.deleteById(id);
    } else {
        throw new RuntimeException("Transaccion identificada con el
ID: " + id + " no encontrado");
    }
}

```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

```

    }

    @PutMapping("/update")
    public void updateTransaccion(@RequestBody Transaccion c) {
        Optional<Transaccion> t = it.findById(c.getId());
        if (!t.equals(null)) {
            it.save(c);
        } else {
            throw new RuntimeException("Transaccion identificada con el
ID: " + c.getId() + " no encontrado");
        }
    }
}

```

### API Banco Datacenter

Ya contemplado el proceso de creación del proyecto, vamos a definir el proceso de configuración y desarrollo de los componentes y módulos requeridos.

#### Application.properties

En este archivo se definen todos los datos relacionados con el puerto por medio del cual estará disponible la API y las configuraciones requeridas para el uso de la base de datos:

```

server.port=3003 // Puerto de acceso
spring.jpa.database=MYSQL // Tipo de SGBD
spring.jpa.hibernate.ddl-auto=Update // Define si la modificación de las entitys
definirá una modificación en la base de datos
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver // nombre del
driver
spring.datasource.url = jdbc:mysql://localhost:3306/
datacenter?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimestC
ode=false&serverTimezone=UTC // URL o PATH de acceso a la base de datos
spring.datasource.username=root // Usuario de la base de datos
spring.datasource.password=password // contraseña del usuario
spring.jpa.open-in-view=true // Visualización de sentencias SQL


```

#### Model

#### Repository

#### Rest

Este paquete se creará en el paquete *co.edu.usbbog.bdd* y será en el cual se definan los métodos a los cuales se podrán tener acceso por medio del mapeo de rutas y el uso de las interfaces para cada uno de los objetos previamente definidas en el paquete repository.

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

### CiudadController

Clase definida con la anotación **RestController** que contiene todos los métodos a los cuales podrán tener acceso mediante la ruta **localhost:3003/ciudad/**, ruta en la cual se podrán realizar las acciones creación, listado, búsqueda por ID, conteo, eliminación y actualización de los registros de ciudades contenidos en la tabla ciudad de la base de datos datacenter, todo esto por medio de la interfaz ICiudad definida para la entidad Ciudad.

```
package co.edu.usbbog.bdd.rest;

import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import co.edu.usbbog.bdd.model.Ciudad;
import co.edu.usbbog.bdd.repository.ICiudad;


@RestController
@RequestMapping("/ciudad")
public class CiudadController {

    @Autowired
    private ICiudad ic;

    @PostMapping("/create")
    public void insertCiudad(@RequestBody Ciudad c) {
        ic.save(c);
    }

    @GetMapping("/all")
    public List<Ciudad> findAllCiudades() {
        List<Ciudad> l = ic.findAll();
        if (l.isEmpty() || l.equals(null)) {
            throw new RuntimeException("No hay ciudades registradas");
        } else {
            return l;
        }
    }

    @GetMapping("/find/{id}")
    public Optional<Ciudad> findCiudad(@PathVariable("id") long id) {
```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	---	----------

```

        Optional<Ciudad> ci = ic.findById(id);
        if (!ci.equals(null)) {
            return ci;
        } else {
            throw new RuntimeException("Ciudad identificada con el ID: "
+ id + " no encontrado");
        }
    }

    @GetMapping("/count")
    public long countCiudades() {
        long c = ic.count();
        if (c != 0) {
            return c;
        } else {
            throw new RuntimeException("No hay ciudades registradas");
        }
    }


    @DeleteMapping("/delete/{id}")
    public void deleteCiudad(@PathVariable("id") long id) {
        Optional<Ciudad> c = ic.findById(id);
        if (c.equals(null)) {
            throw new RuntimeException("Ciudad identificada con el ID: "
+ id + " no encontrado");
        } else {
            ic.deleteById(id);
        }
    }

    @PutMapping("/update")
    public void updateCiudad(@RequestBody Ciudad c) {
        Optional<Ciudad> ci = ic.findById(c.getId());
        if (ci.equals(null)) {
            throw new RuntimeException("Ciudad identificada con el ID: "
+ c.getId() + " no encontrado");
        } else {
            ic.save(c);
        }
    }
}

```

### CuentaController

Clase definida con la anotación **RestController** que contiene todos los métodos a los cuales podrán tener acceso mediante la ruta **localhost:3003/cuenta/**, ruta en la cual se podrán realizar las acciones creación, listado, búsqueda por ID, conteo, eliminación y actualización de los registros de cuentas contenidas en la tabla cuenta

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

de la base de datos datacenter, todo esto por medio de la interfaz ICuenta definida para la entidad Cuenta.

```
package co.edu.usbbog.bdd.rest;


import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import co.edu.usbbog.bdd.model.Cuenta;
import co.edu.usbbog.bdd.repository.ICuenta;

@RestController
@RequestMapping("/cuenta")
public class CuentaController {
    @Autowired
    private ICuenta ic;

    @PostMapping("/create")
    public void insertCuenta(@RequestBody Cuenta c) {
        ic.save(c);
    }

    @GetMapping("/all")
    public List<Cuenta> findAllCuentas() {
        List<Cuenta> l = ic.findAll();
        if (l.isEmpty() || l.equals(null)) {
            throw new RuntimeException("No hay cuentas registradas");
        } else {
            return l;
        }
    }

    @GetMapping("/find/{id}")
    public Optional<Cuenta> findCuenta(@PathVariable("id") long id) {
        Optional<Cuenta> cu = ic.findById(id);
        if (!cu.equals(null)) {
            return cu;
        } else {
            throw new RuntimeException("Cuenta identificada con el ID: "
+ id + " no encontrado");
        }
    }
}
```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	---	----------

```

    }

    @GetMapping("/count")
    public long countCuenta() {
        long c = ic.count();
        if (c != 0) {
            return c;
        } else {
            throw new RuntimeException("No hay cuentas registradas");
        }
    }

    @DeleteMapping("/delete/{id}")
    public void deleteCuenta(@PathVariable("id") long id) {
        Optional<Cuenta> cu = ic.findById(id);
        if (!cu.equals(null)) {
            ic.deleteById(id);
        } else {
            throw new RuntimeException("Cuenta identificada con el ID: "
+ id + " no encontrado");
        }
    }

    @PutMapping("/update")
    public void updateCuenta(@RequestBody Cuenta c) {
        Optional<Cuenta> cu = ic.findById(c.getNum());
        if (!cu.equals(null)) {
            ic.save(c);
        } else {
            throw new RuntimeException("Cuenta identificada con el ID: "
+ c.getNum() + " no encontrado");
        }
    }
}

```

#### TipoTransaccionController

Clase definida con la anotación **RestController** que contiene todos los métodos a los cuales podrán tener acceso mediante la ruta **localhost:3003/tipotransaccion/**, ruta en la cual se podrán realizar las acciones creación, listado, búsqueda por ID, conteo, eliminación y actualización de los registros de tipos de transacciones contenidos en la tabla tipotransaccion de la base de datos datacenter, todo esto por medio de la interfaz ITipoTransaccion definida para la entidad TipoTransaccion.


```

package co.edu.usbbog.bdd.rest;

import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;

```



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import co.edu.usbbog.bdd.model.TipoTransaccion;
import co.edu.usbbog.bdd.repository.ITipoTransaccion;

@RestController
@RequestMapping("/tipotransaccion")
public class TipoTransaccionController {
    @Autowired
    private ITipoTransaccion itt;


    @PostMapping("/create")
    public void insertTipoTransaccion(@RequestBody TipoTransaccion tt) {
        itt.save(tt);
    }

    @GetMapping("/all")
    public List<TipoTransaccion> findAllTipoTransacciones() {
        List<TipoTransaccion> l = itt.findAll();
        if (l.isEmpty() || l.equals(null)) {
            throw new RuntimeException("No hay tipos de transacciones
registradas");
        } else {
            return l;
        }
    }

    @GetMapping("/find/{id}")
    public Optional<TipoTransaccion> findTipoTransaccion(@PathVariable("id")
long id) {
        Optional<TipoTransaccion> tt = itt.findById(id);
        if (!tt.equals(null)) {
            return tt;
        } else {
            throw new RuntimeException("Tipo de Tramsaccion identificada
con el ID: " + id + " no encontrado");
        }
    }

    @GetMapping("/count")
    public long countTipoTransacciones() {
        long c = itt.count();
        if (c != 0) {
            return c;
        }
    }
}

```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	---	----------

```

    } else {
        throw new RuntimeException("No hay tipos de transacciones
registradas");
    }
}

@DeleteMapping("/delete/{id}")
public void deleteTipoTransaccion(@PathVariable("id") long id) {
    Optional<TipoTransaccion> tt = itt.findById(id);
    if (!tt.equals(null)) {
        itt.deleteById(id);
    } else {
        throw new RuntimeException("Tipo de Tramsaccion identificada
con el ID: " + id + " no encontrado");
    }
}

@PutMapping("/update")
public void updateTipoTransaccion(@RequestBody TipoTransaccion tt) {
    Optional<TipoTransaccion> t = itt.findById(tt.getId());
    if (!t.equals(null)) {
        itt.save(tt);
    } else {
        throw new RuntimeException("Tipo de Tramsaccion identificada
con el ID: " + tt.getId() + " no encontrado");
    }
}
}

```

#### AuditoriaController


Clase definida con la anotación **RestController** que contiene todos los métodos a los cuales podrán tener acceso mediante la ruta **localhost:3003/auditoria/**, ruta en la cual se podrán realizar las acciones creación, listado, búsqueda por ID, conteo, eliminación y actualización de los registros de auditorios contenidos en la tabla auditoria de la base de datos datacenter, todo esto por medio de la interfaz **IAuditoria** definida para la entidad Auditoria.

```

package co.edu.usbbog.bdd.rest;

import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;

```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```
import org.springframework.web.bind.annotation.RestController;
import co.edu.usbbog.bdd.model.Auditoria;
import co.edu.usbbog.bdd.repository.IAuditoria;

@RestController
@RequestMapping("/auditoria")
public class AuditoriaController {

    @Autowired
    IAuditoria ia;


    @PostMapping("/create")
    public void insertTransaccion(@RequestBody Auditoria a) {
        ia.save(a);
    }

    @GetMapping("/all")
    public List<Auditoria> findAllTransacciones() {
        List<Auditoria> l = ia.findAll();
        if (l.isEmpty() || l.equals(null)) {
            throw new RuntimeException("No hay auditorias registradas");
        } else {
            return l;
        }
    }

    @GetMapping("/find/{id}")
    public Optional<Auditoria> findTransaccion(@PathVariable("id") long id) {
        Optional<Auditoria> t = ia.findById(id);
        if (!t.equals(null)) {
            return t;
        } else {
            throw new RuntimeException("Auditoria identificada con el ID:
" + id + " no encontrado");
        }
    }

    @GetMapping("/count")
    public long countTransaccion() {
        long c = ia.count();
        if (c != 0) {
            return c;
        } else {
            throw new RuntimeException("No hay auditorias registradas");
        }
    }

    @DeleteMapping("/delete/{id}")
    public void deleteTransaccion(@PathVariable("id") long id) {
```

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	---	----------

```

Optional<Auditoria> t = ia.findById(id);
if (!t.equals(null)) {
    ia.deleteById(id);
} else {
    throw new RuntimeException("Tramsaccion identificada con el
ID: " + id + " no encontrado");
}
}

@PutMapping("/update")
public void updateTransaccion(@RequestBody Auditoria a) {
    Optional<Auditoria> t = ia.findById(a.getId());
    if (!t.equals(null)) {
        ia.save(a);
    } else {
        throw new RuntimeException("Tramsaccion identificada con el
ID: " + a.getId() + " no encontrado");
    }
}
}

```


## Pruebas

Para la realización de pruebas nos dirigimos a Postman, en donde podremos crear Requests o peticiones bajo los paths definidos para cada uno de los métodos contenidos en las clases Rest Controller de cada entidad. De modo tal, se demostrarán los Requests creados para las entidades Ciudad, TipoTransaccion, Cuenta, transacción y Auditoria, partiendo del previo conocimiento que las entidades Ciudad, TipoTransaccion y Cuenta hacen parte de las tres APIs, compartiendo su estructura y funcionalidad, y conociendo que la entidad Transacción solo pertenece a las APIS referentes a las bases de datos de las sedes del banco en las dos ciudades, y la entidad Auditoria solo se encontrara en la API referente al datacenter.

## Requests Ciudad

Partiendo de la creación del REST Controller en las tres APIs y teniendo en cuenta de que el funcionamiento de este REST Controller es análogo en todas las APIS, se tendrá presente que el acceso a este REST Controller se realizará mediante el siguiente path:

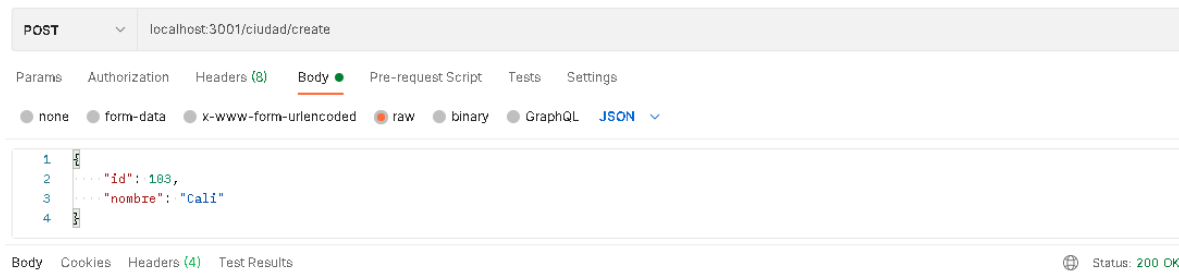
API	Path
api.banco_bogota	localhost:3001/ciudad
api.banco_medellin	localhost:3002/ciudad
api.datacenter	localhost:3003/ciudad

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
--	---	---	-----------------

A continuación, se expondrán las pruebas a los 6 métodos implementados en este REST Controller:

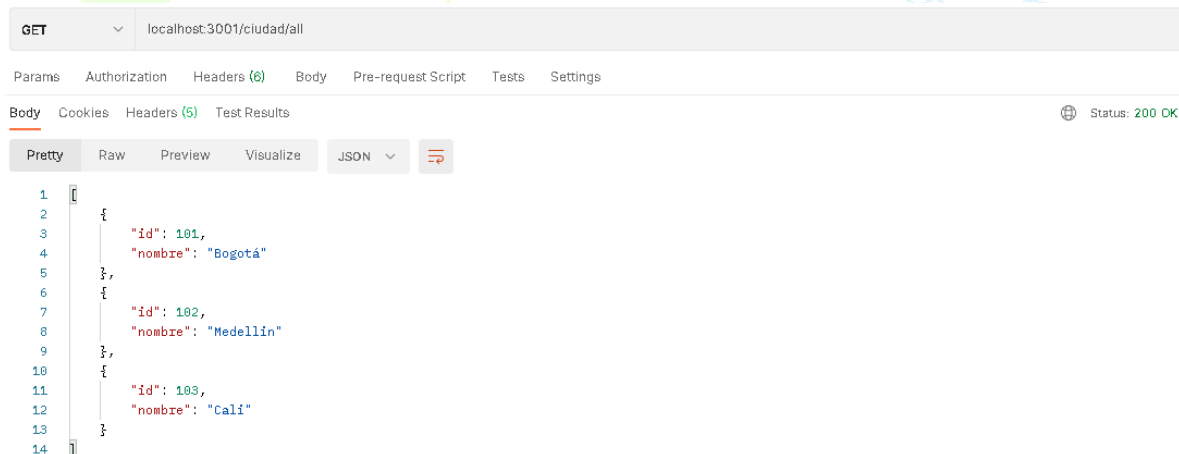
### Create

Método POST que recibe en su cuerpo/body un JSON que cuenta con la estructura de la entidad Ciudad, y retornara como estado 200 si se pudo realizar la correcta inserción del registro en la base de datos:




### Find All

Método GET sin parámetros de entrada, el cual retornara una lista en formato JSON con todos los registros de la tabla Ciudad:



### Find by ID

Método GET que recibirá a través del path/url el id de la ciudad requerida por el usuario:

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

GET localhost:3001/ciudad/find/103

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": 103,
3   "nombre": "Cali"
4 }
```

### Count

Método GET sin parámetros de entrada que retornara un valor numérico con la cantidad de registros en la tabla Ciudad:

GET localhost:3001/ciudad/count

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```

1 1
```

### Update

Método PUT que recibirá en el cuerpo/body de la petición un objeto JSON que hace referencia a la Entidad ciudad, basándose en el ID entregado dentro del objeto para realizar la actualización del registro en la base de datos:

PUT localhost:3001/ciudad/update

Params Authorization Headers (6) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "id": 103,
3   "nombre": "Calip"
4 }
```

Body Cookies Headers (4) Test Results Status: 200 OK


### Delete

Método DELETE que recibirá un parámetro de entrada en el PATH/URL de acceso que hace referencia al ID del registro de la ciudad que requiere eliminar:

DELETE localhost:3001/ciudad/delete/103

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (4) Test Results Status: 200 OK

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
--	---	---	----------

### Tabla de métodos


A continuación, se verán representados los Paths para el acceso a cada uno de los métodos contenidos en este REST Controller y los elementos requeridos en el cuerpo de cada petición:

Nombre método	Path	Body / Json Ejemplo
Create	localhost:puerto_API/ciudad/create	<pre>{   "id":103,   "nombre":"Cali" }</pre>
Find all	localhost:Puerto_API/ciudad/all	-----
Find by ID	localhost:Puerto_API/ciudad/find/*id*	-----
Count	localhost: Puerto_API /ciudad/count	-----
Update	localhost:Puerto_API/ciudad/update	<pre>{   "id":103,   "nombre":"Cali" }</pre>
Delete	localhost:Puerto_API/ciudad/delete/*id*	-----

### Requests TipoTransaccion

Partiendo de la creación del REST Controller en las tres APIs y teniendo en cuenta de que el funcionamiento de este REST Controller es análogo en todas las APIS, se tendrá presente que el acceso a este REST Controller se realizará mediante el siguiente path:

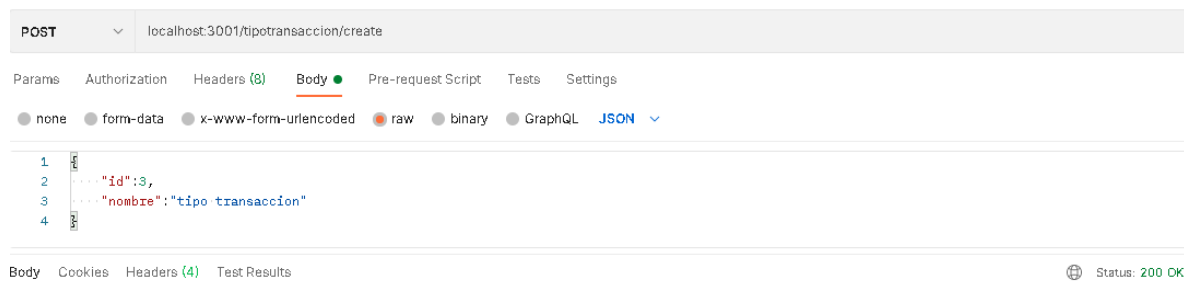
API	Path
api.banco_bogota	localhost:3001/tipotransaccion
api.banco_medellin	localhost:3002/ tipotransaccion
api.datacenter	localhost:3003/ tipotransaccion

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
--	---	---	-----------------

A continuación, se expondrán las pruebas a los 6 métodos implementados en este REST Controller:

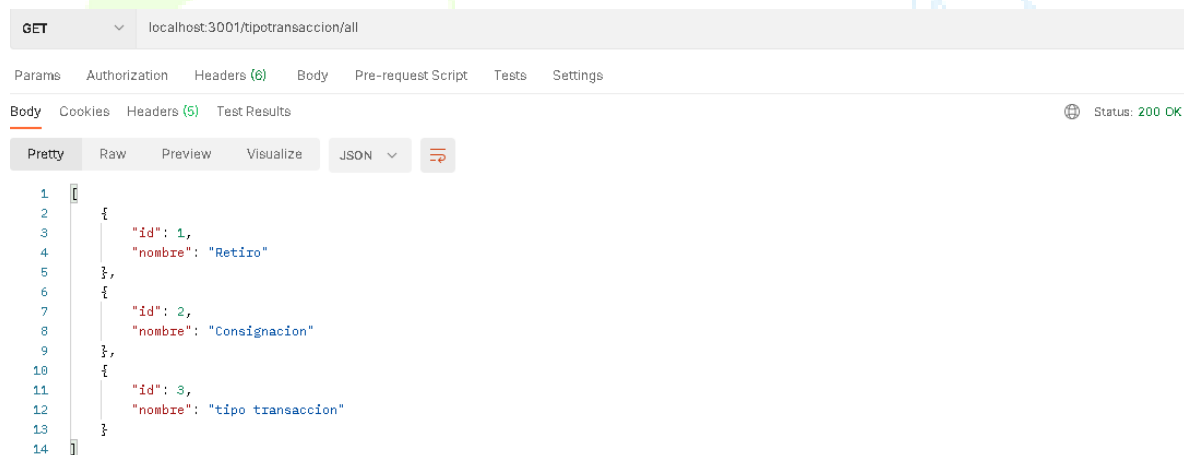
### Create

Método POST que recibe en su cuerpo/body un JSON que cuenta con la estructura de la entidad TipoTransaccion, y retornara como estado 200 si se pudo realizar la correcta inserción del registro en la base de datos:



### Find All


Método GET sin parámetros de entrada, el cual retornara una lista en formato JSON con todos los registros de la tabla TipoTransaccion:



### Find by ID

Método GET que recibirá a través del path/url el id del tipo de transacción requerido por el usuario:



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

GET localhost:3001/tipotransaccion/find/3

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": 3,
3   "nombre": "tipo transaccion"
4 }
```

### Count

Método GET sin parámetros de entrada que retornara un valor numérico con la cantidad de registros en la tabla TipoTransaccion:

GET localhost:3001/tipotransaccion/count

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```

1 3
```

### Update

Método PUT que recibirá en el cuerpo/body de la petición un objeto JSON que hace referencia a la Entidad TipoTransaccion, basándose en el ID entregado dentro del objeto para realizar la actualización del registro en la base de datos:

PUT localhost:3001/tipotransaccion/update

Params Authorization Headers (6) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "id": 3,
3   "nombre": "tipo transaccion 3"
4 }
```

Body Cookies Headers (4) Test Results Status: 200 OK


### Delete

Método DELETE que recibirá un parámetro de entrada en el PATH/URL de acceso que hace referencia al ID del registro del tipo de transacción que requiere eliminar:

DELETE localhost:3001/tipotransaccion/delete/3

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (4) Test Results Status: 200 OK

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------

### Tabla de métodos


A continuación, se verán representados los Paths para el acceso a cada uno de los métodos contenidos en este REST Controller y los elementos requeridos en el cuerpo de cada petición:

Nombre método	Path	Body / Json Ejemplo
Create	localhost:puerto_API/tipotransaccion/create	<pre>{   "id":3,   "nombre":"tipo transaccion" }</pre>
Find all	localhost:Puerto_API/tipotransaccion /all	-----
Find by ID	localhost:Puerto_API/tipotransaccion /find/*id*	-----
Count	localhost: Puerto_API / tipotransaccion /count	-----
Update	localhost:Puerto_API/tipotransaccion /update	<pre>{   "id":3,   "nombre":"tipo transaccion" }</pre>
Delete	localhost:Puerto_API/tipotransaccion /delete/*id*	-----

### Requests Cuenta

Partiendo de la creación del REST Controller en las tres APIs y teniendo en cuenta de que el funcionamiento de este REST Controller es análogo en todas las APIS, se tendrá presente que el acceso a este REST Controller se realizará mediante el siguiente path:

API	Path
api.banco_bogota	localhost:3001/cuenta
api.banco_medellin	localhost:3002/ cuenta

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
--	---	---	-----------------

api.datacenter

localhost:3003/ cuenta

**Nota:** aunque la Entidad/Tabla de la API datacenter tiene una estructura diferente, su funcionamiento se mantiene análogo al de las demás APIs, debido a que comparten el mismo tipo de Clave primaria.

A continuación, se expondrán las pruebas a los 6 métodos implementados en este REST Controller:

### Create

Método POST que recibe en su cuerpo/body un JSON que cuenta con la estructura de la entidad Cuenta, y retornara como estado 200 si se pudo realizar la correcta inserción del registro en la base de datos:

API banco Bogota / Insert Cuenta

POST localhost:3001/cuenta/create

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "num": 382,
3   "cedula": 12345,
4   "nombre": "Felipe",
5   "telefono": "3144432469",
6   "saldo": 1000.00,
7   "ciudad": {
8     "id": 102,
9     "nombre": "Medellin"
10  }
11 }

```

Body Cookies Headers (4) Test Results

Status: 200 OK

### Find All

Método GET sin parámetros de entrada, el cual retornara una lista en formato JSON con todos los registros de la tabla Cuenta:

GET localhost:3001/cuenta/all

Params Authorization Headers (8) Body Pre-request Script Tests Settings


Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```

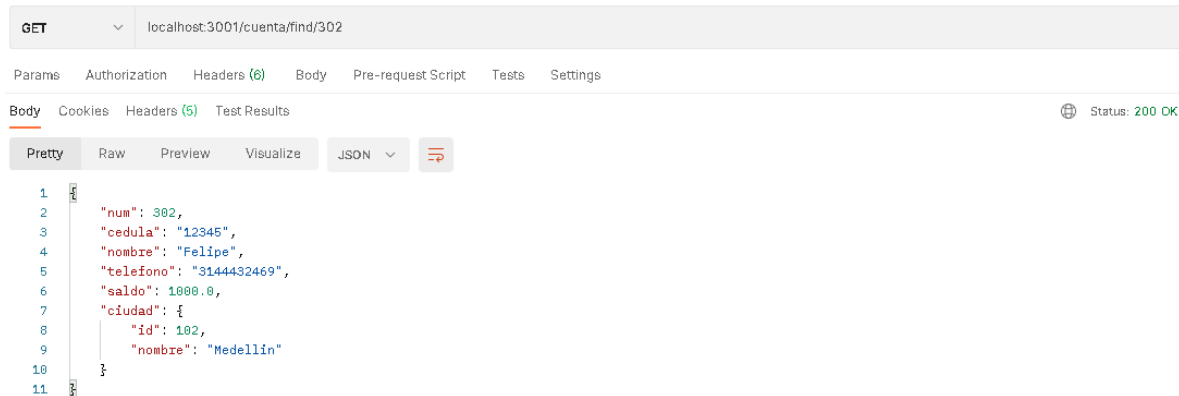
1 [
2   {
3     "num": 382,
4     "cedula": "12345",
5     "nombre": "Felipe",
6     "telefono": "3144432469",
7     "saldo": 1000.0,
8     "ciudad": {
9       "id": 102,
10      "nombre": "Medellin"
11    }
12  }
13 ]

```

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
--	---	---	-----------------

### Find by ID

Método GET que recibirá a través del path/url el id de la Cuenta requerida por el usuario:



```

GET localhost:3001/cuenta/find/302

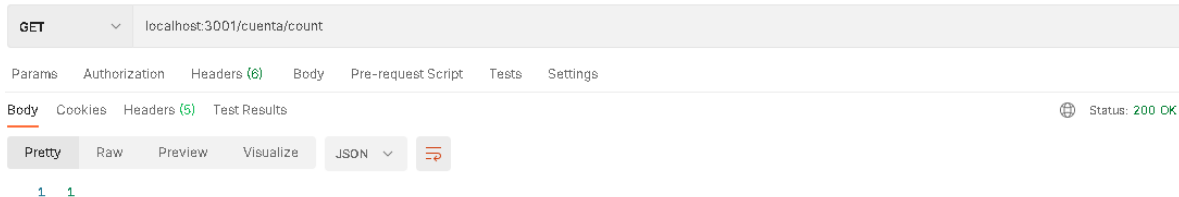
Body
  Cookies Headers (5) Test Results
  Status: 200 OK

Pretty Raw Preview Visualize JSON
{
  "num": 302,
  "cedula": "12345",
  "nombre": "Felipe",
  "telefono": "3144432469",
  "saldo": 1000.0,
  "ciudad": {
    "id": 102,
    "nombre": "Medellin"
  }
}

```

### Count

Método GET sin parámetros de entrada que retornara un valor numérico con la cantidad de registros en la tabla Cuenta:



```

GET localhost:3001/cuenta/count


Body
  Cookies Headers (5) Test Results
  Status: 200 OK

Pretty Raw Preview Visualize JSON
{
  "count": 1
}

```

### Update

Método PUT que recibirá en el cuerpo/body de la petición un objeto JSON que hace referencia a la Entidad Cuenta, basándose en el ID entregado dentro del objeto para realizar la actualización del registro en la base de datos:

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

PUT localhost:3001/cuenta/update

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "num": 302,
3   "cedula": "54321",
4   "nombre": "Luis Velasco",
5   "telefono": "3144432469",
6   "saldo": 1000.00,
7   "ciudad": {
8     "id": 101,
9     "nombre": "Medellin"
10  }
11 }

```

Body Cookies Headers (4) Test Results

Status: 200 OK

## Delete

Método DELETE que recibirá un parámetro de entrada en el PATH/URL de acceso que hace referencia al ID del registro de la Cuenta que requiere eliminar:

DELETE localhost:3001/cuenta/delete/302

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (4) Test Results

Status: 500 Internal Server Error Time: 282 ms Size: 11.76 KB Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "timestamp": "2021-05-31T22:37:10.692+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "trace": "org.springframework.dao.DataIntegrityViolationException: could not execute statement; SQL [n/a]; constraint [null]; nested exception is org.hibernate.exception."
6 }

```


No permite la eliminación de la cuenta ya que existen transacciones asociadas a la clave primaria de este registro:

[java.sql.SQLIntegrityConstraintViolationException](#): Cannot delete or update a parent row: a foreign key constraint fails (`banco\_bogota`.`transaccion`, CONSTRAINT `fk\_transaccion\_cuenta` FOREIGN KEY (`cuenta`) REFERENCES `cuenta` (`num`) ON DELETE NO ACTION ON UPDATE NO ACTION)

## Tabla de métodos

A continuación, se verán representados los Paths para el acceso a cada uno de los métodos contenidos en este REST Controller y los elementos requeridos en el cuerpo de cada petición:

Nombre método	Path	Body / Json Ejemplo
Create	localhost:puerto_API/cuenta/create	<pre> {   "num": 302,   "cedula": 12345,   "nombre": "Felipe",   "telefono": "3144432469", </pre>

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------

```
"saldo":1000.00,
"ciudad":{
  "id":102,
  "nombre":"Medellin"
}
}
```

**Para la API del  
datacenter:**

```
{
  "num":302,
  "saldo":1000.00,
}
```

Find all localhost:Puerto\_API/  
cuenta /all

-----

Find by ID localhost:Puerto\_API/  
cuenta /find/\*id\*

-----

Count localhost: Puerto\_API /  
tipotransaccion /count


-----

Update localhost:Puerto\_API/  
cuenta /update

```
{
  "num":302,
  "cedula":12345,
  "nombre":"Felipe",
  "telefono":"3144432469",
  "saldo":1000.00,
  "ciudad":{
    "id":102,
    "nombre":"Medellin"
  }
}
```

**Para la API del  
datacenter:**

```
{
  "num":302,
  "saldo":1000.00,
}
```

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
---	---	---	-----------------

Delete localhost:Puerto\_API/ -----  
cuenta /delete/\*id\*

## Requests Transaccion

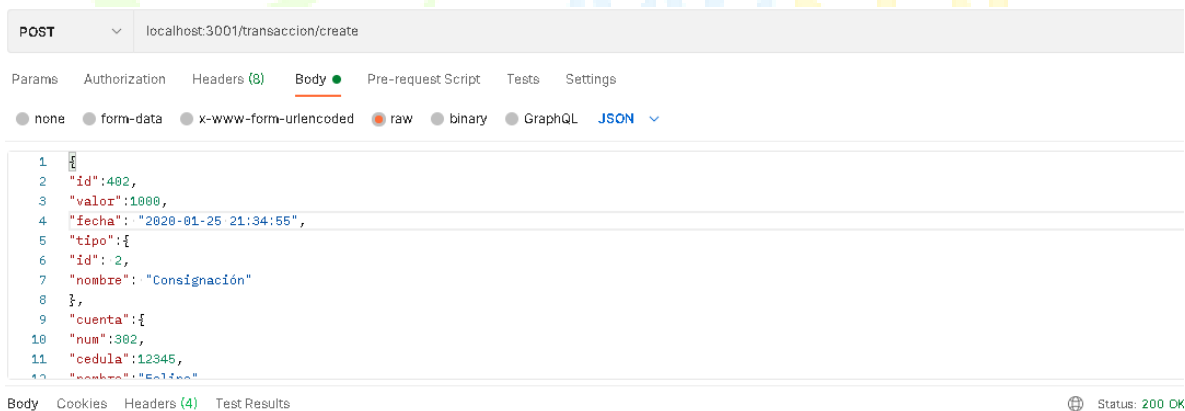
Partiendo de la creación del REST Controller en las dos APIs referentes a las sedes en las dos ciudades del banco y teniendo en cuenta de que el funcionamiento de este REST Controller es análogo en todas las APIs, se tendrá presente que el acceso a este REST Controller se realizará mediante el siguiente path:

API	Path
api.banco_bogota	localhost:3001/transaccion
api.banco_medellin	localhost:3002/ transaccion

A continuación, se expondrán las pruebas a los 6 métodos implementados en este REST Controller:

### Create


Método POST que recibe en su cuerpo/body un JSON que cuenta con la estructura de la entidad Transaccion, y retornara como estado 200 si se pudo realizar la correcta inserción del registro en la base de datos:



Tener en cuenta el formato del atributo fecha, el cual debe estar definido de tal forma concuerde con el JsonFormatter definido.

### Find All

Método GET sin parámetros de entrada, el cual retornara una lista en formato JSON con todos los registros de la tabla Transaccion:

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

```

GET localhost:3001/transaction/all

Body
Headers (5)
Test Results
Status: 200 OK

Pretty
Raw
Preview
Visualize
JSON
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
{
  "id": 401,
  "valor": 1000.0,
  "fecha": "2016-01-25 21:34:55",
  "tipo": {
    "id": 1,
    "nombre": "Retiro"
  },
  "cuenta": {
    "num": 302,
    "cedula": "54321",
    "nombre": "Luis Velasco",
    "telefono": "3144432469",
    "saldo": 1000.0,
    "ciudad": {
      "id": 101,
      "nombre": "Bogotá"
    }
  }
},
{
  "id": 402,
  "valor": 1000.0,
  "fecha": "2020-01-25 21:34:55",
  "tipo": {
    "id": 2,
    "nombre": "Consignacion"
  },
  "cuenta": {
    "num": 302,
    "cedula": "54321",
    "nombre": "Luis Velasco",
    "telefono": "3144432469",
    "saldo": 1000.0,
    "ciudad": {
      "id": 101,

```

### Find by ID

Método GET que recibirá a través del path/url el id de la Transaccion requerida por el usuario:

```

GET localhost:3001/transaction/find/401


Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Status: 200 OK

Body
Cookies
Headers (5)
Test Results

Pretty
Raw
Preview
Visualize
JSON
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
{
  "id": 401,
  "valor": 1000.0,
  "fecha": "2016-01-25 21:34:55",
  "tipo": {
    "id": 1,
    "nombre": "Retiro"
  },
  "cuenta": {
    "num": 302,
    "cedula": "54321",
    "nombre": "Luis Velasco",
    "telefono": "3144432469",
    "saldo": 1000.0,
    "ciudad": {
      "id": 101,
      "nombre": "Bogotá"
    }
  }
}

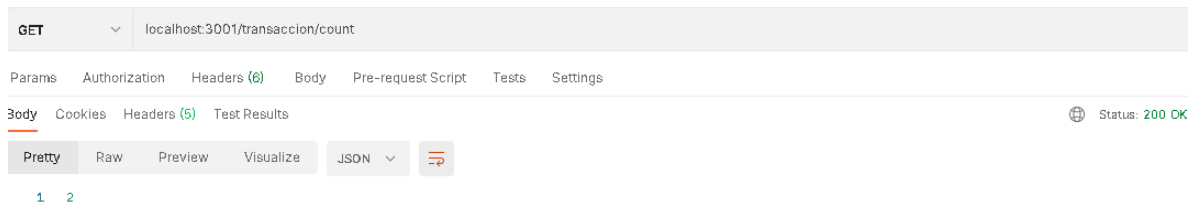
```



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
--	---	---	-----------------

### Count

Método GET sin parámetros de entrada que retornara un valor numérico con la cantidad de registros en la tabla Transaccion:



GET localhost:3001/transaccion/count

Params Authorization Headers (6) Body Pre-request Script Tests Settings

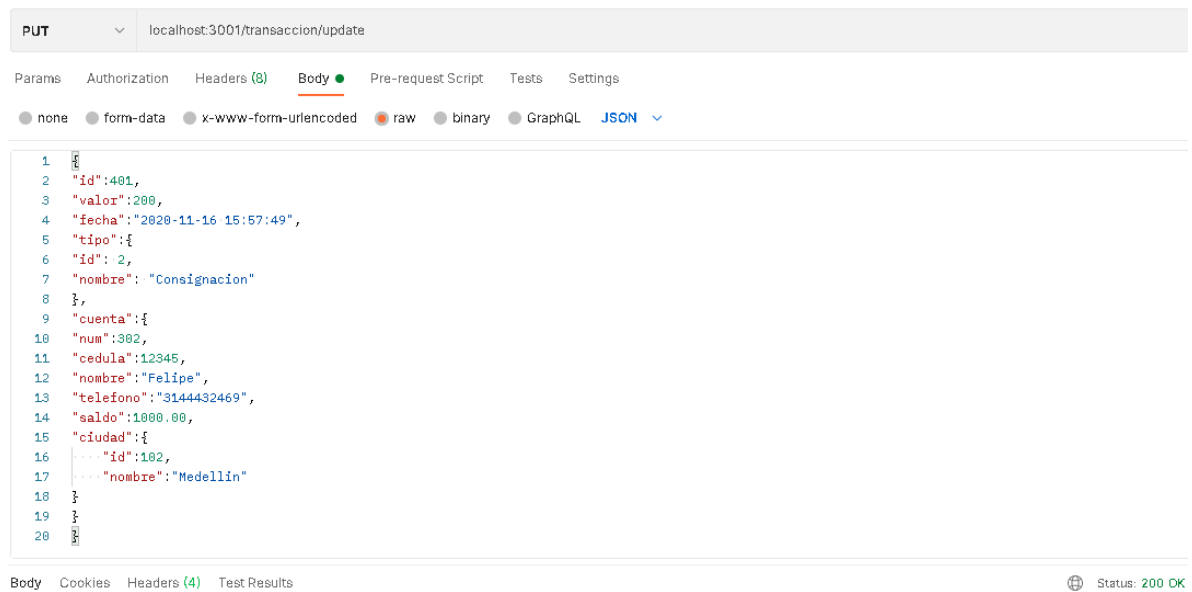
Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 2
```

### Update

Método PUT que recibirá en el cuerpo/body de la petición un objeto JSON que hace referencia a la Entidad Transaccion, basándose en el ID entregado dentro del objeto para realizar la actualización del registro en la base de datos:



PUT localhost:3001/transaccion/update

Params Authorization Headers (8) Body Pre-request Script Tests Settings

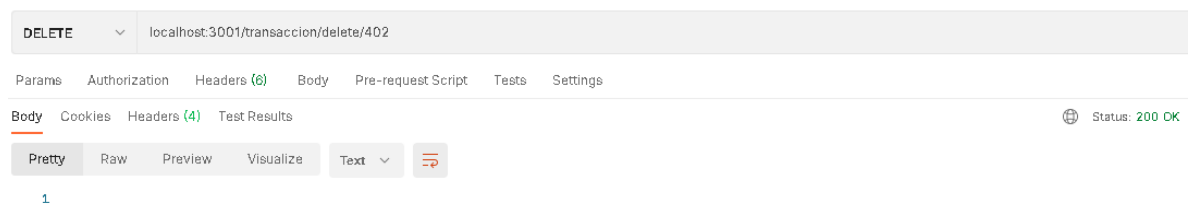
none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "id": 401,
3   "valor": 200,
4   "fecha": "2020-11-16 15:57:49",
5   "tipo": {
6     "id": 2,
7     "nombre": "Consignacion"
8   },
9   "cuenta": {
10    "num": 302,
11    "cedula": 12345,
12    "nombre": "Felipe",
13    "telefono": "3144432469",
14    "saldo": 1000.00,
15    "ciudad": {
16      "id": 102,
17      "nombre": "Medellin"
18    }
19  }
20 }
```

Body Cookies Headers (4) Test Results Status: 200 OK

### Delete

Método DELETE que recibirá un parámetro de entrada en el PATH/URL de acceso que hace referencia al ID del registro de la transacción que requiere eliminar:




DELETE localhost:3001/transaccion/delete/402

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (4) Test Results Status: 200 OK

Pretty Raw Preview Visualize Text


```
1
```

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------

### *Tabla de métodos*

A continuación, se verán representados los Paths para el acceso a cada uno de los métodos contenidos en este REST Controller y los elementos requeridos en el cuerpo de cada petición:

Nombre método	Path	Body / Json Ejemplo
Create	localhost:puerto_API/transaccion/create	<pre>{   "id":401,   "valor":1000,   "fecha":"2020-11-16 15:57:49",   "tipo":{     "id": 1,     "nombre": "Retiro"   },   "cuenta":{     "num":302,     "cedula":12345,     "nombre":"Felipe",     "telefono":"3144432469",     "saldo":1000.00,     "ciudad":{       "id":102,       "nombre":"Medellin"     }   } }</pre>
Find all	localhost:Puerto_API/transaccion /all	-----
Find by ID	localhost:Puerto_API/transaccion /find/*id*	-----
Count	localhost: Puerto_API / transaccion /count	-----
Update	localhost:Puerto_API/transaccion /update	<pre>{   "id":401,   "valor":1000,   "fecha":"2020-11-16 15:57:49",</pre>

 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------

```

"tipo":{
  "id": 1,
  "nombre": "Retiro"
},
"cuenta":{
  "num":302,
  "cedula":12345,
  "nombre":"Felipe",
  "telefono":"3144432469",
  "saldo":1000.00,
  "ciudad":{
    "id":102,
    "nombre":"Medellin"
  }
}
}
}

```

Delete      localhost:Puerto\_API/  
transaccion /delete/\*id\*

### Requests Auditoria


Teniendo en cuenta que este REST Controller debe ya estar implementado en la API referente al datacenter, al igual que la base de datos de esta API, se definirá el siguiente path de acceso a los métodos contenidos en este REST Controller:

API	Path
api.datacenter	localhost:3001/auditoria

A continuación, se expondrán las pruebas a los 6 métodos implementados en este REST Controller:

### Create

Método POST que recibe en su cuerpo/body un JSON que cuenta con la estructura de la entidad Auditoria, y retornara como estado 200 si se pudo realizar la correcta inserción del registro en la base de datos:

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

POST localhost:3003/auditoria/create

Paramere Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

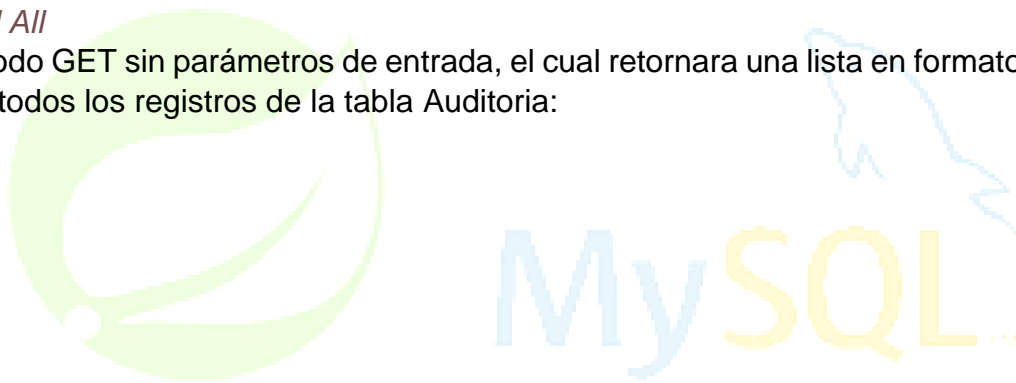
1 {
2   "id": 402,
3   "valor": 1000,
4   "fecha": "2020-12-16 15:57:49",
5   "tipo": {
6     "id": 1,
7     "nombre": "Retiro"
8   },
9   "cuenta": {
10    "num": 302,
11    "cedula": 12345,
12    "nombre": "Felipe",
13    "telefono": "3144432469",
14    "saldo": 1000.00,
15    "ciudad": {
16      "id": 102,
17      "nombre": "Medellín"
18    },
19  },
20  "ciudad": {
21    "id": 102,
22    "nombre": "Medellín"
23  },
24 }


```

Body Cookies Headers (4) Test Results Status: 200 OK

### Find All

Método GET sin parámetros de entrada, el cual retornara una lista en formato JSON con todos los registros de la tabla Auditoria:



 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

GET localhost:3003/auditoria/all

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON


```

1  [
2    {
3      "id": 401,
4      "valor": 1000.0,
5      "fecha": "2020-11-16 15:57:49",
6      "tipo": {
7        "id": 1,
8        "nombre": "Retiro"
9      },
10     "cuenta": {
11       "num": 302,
12       "saldo": 1000.0
13     },
14     "ciudad": {
15       "id": 102,
16       "nombre": "Medellin"
17     }
18   },
19   {
20     "id": 402,
21     "valor": 1000.0,
22     "fecha": "2020-12-16 15:57:49",
23     "tipo": {
24       "id": 1,
25       "nombre": "Retiro"
26     },
27     "cuenta": {
28       "num": 302,
29       "saldo": 1000.0
30     },
31     "ciudad": {
32       "id": 102,
33       "nombre": "Medellin"
34     }
35   }
36 ]

```

### Find by ID

Método GET que recibirá a través del path/url el id de la Auditoria requerida por el usuario:

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
--	---	---	-----------------

GET localhost:3003/auditoria/find/401

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```

1  {
2    "id": 401,
3    "valor": 1000.0,
4    "fecha": "2020-11-16 15:57:49",
5    "tipo": {
6      "id": 1,
7      "nombre": "Retiro"
8    },
9    "cuenta": {
10     "num": 302,
11     "saldo": 1000.0
12   },
13   "ciudad": {
14     "id": 102,
15     "nombre": "Medellín"
16   }
17 }

```

### Count

Método GET sin parámetros de entrada que retornara un valor numérico con la cantidad de registros en la tabla Auditoria:

GET localhost:3003/auditoria/count

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON


```

1  2

```

### Update

Método PUT que recibirá en el cuerpo/body de la petición un objeto JSON que hace referencia a la Entidad Auditoria, basándose en el ID entregado dentro del objeto para realizar la actualización del registro en la base de datos:

 <p>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</p>	<p>Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas</p>	<p>Desarrollo de APIS – Ejercicio del Banco</p>	<p>2021 - 2</p>
---	---	---	-----------------

PUT localhost:3003/auditoria/update

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "id": 402,
3   "valor": 2000,
4   "fecha": "2020-12-17 15:57:49",
5   "tipo": {
6     "id": 2,
7     "nombre": "Consignacion"
8   },
9   "cuenta": {
10    "num": 302,
11    "cedula": 12345,
12    "nombre": "Felipe",
13    "telefono": "3144432469",
14    "saldo": 1000.00,
15    "ciudad": {
16      "id": 102,
17      "nombre": "Medellin"
18    }
19  },
20  "ciudad": {
21    "id": 102,
22    "nombre": "Medellin"
23  }
24 }
```

Body Cookies Headers (4) Test Results Status: 200 OK

### Delete

Método DELETE que recibirá un parámetro de entrada en el PATH/URL de acceso que hace referencia al ID del registro de la Auditoria que requiere eliminar:

DELETE localhost:3003/auditoria/delete/402


Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (4) Test Results Status: 200 OK

### Tabla de métodos


A continuación, se verán representados los Paths para el acceso a cada uno de los métodos contenidos en este REST Controller y los elementos requeridos en el cuerpo de cada petición:

Nombre método	Path	Body / Json Ejemplo
Create	localhost:puerto_API/auditoria/create	<pre>{   "id": 401,   "valor": 1000,   "fecha": "2020-11-16 15:57:49",   "tipo": {     "id": 1,     "nombre": "Retiro"   } }</pre>

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	---	----------

		<pre> }, "cuenta":{ "num":302, "cedula":12345, "nombre":"Felipe", "telefono":"3144432469", "saldo":1000.00, "ciudad":{ "id":102, "nombre":"Medellin" } }, "ciudad":{ "id":102, "nombre":"Medellin" } } </pre>
Find all	localhost:Puerto_API/ auditoria /all	-----
Find by ID	localhost:Puerto_API/ auditoria /find/*id*	-----
Count	localhost: Puerto_API / auditoria /count	-----
Update	localhost:Puerto_API/ auditoria /update	<pre> { "id":401, "valor":1000, "fecha":"2020-11-16 15:57:49", "tipo":{ "id": 1, "nombre": "Retiro" }, "cuenta":{ "num":302, "cedula":12345, "nombre":"Felipe", "telefono":"3144432469", "saldo":1000.00, "ciudad":{ "id":102, "nombre":"Medellin" } } </pre>



 <b>UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ</b>	<b>Universidad de San Buenaventura</b> <b>Facultad de ingeniería</b> <b>Bases de datos distribuidas</b>	<b>Desarrollo de APIS – Ejercicio del Banco</b>	<b>2021 - 2</b>
--	---	---	-----------------

```

    }
  },
  "ciudad": {
    "id": 102,
    "nombre": "Medellin"
  }
}

```

Delete      localhost:Puerto\_API/  
auditoria /delete/\*id\*      -----

## Conclusiones

En base a todo el desarrollo de la actividad, analizando cada uno de los apartados desarrollados relacionados con el desarrollo de APIS para una base de datos distribuida y teniendo en cuenta las tecnologías utilizadas se dan con las siguientes conclusiones:


- La distribución de una base de datos centralizada siempre se dará teniendo en cuenta las necesidades relacionadas con los datos requeridos en cada uno de los lugares donde se ubicará la distribución de la base de datos, teniendo presente que se debe contemplar el desarrollo de un componente (API o procedimientos de manipulación en lenguaje SQL) que mantenga transparente la distribución de la base de datos y las operaciones realizadas para la gestión de los datos.
- El desarrollo de APIs por medio de Spring Boot facilita mucho el proceso asociando fácilmente por medio de la correcta configuración del servicio, de las dependencias requeridas y de las anotaciones empleadas.
- El uso de Postman facilita demasiado el proceso de pruebas individuales para cada API, comprendiendo también el previo conocimiento de manipulación e objetos tipo JSON, los cuales son de vital importancia en las pruebas de las peticiones a las APIs.

## Anexos

### Repositorios de APIs desarrolladas

A continuación, podrá encontrar las URL que le permitirán el acceso a cada uno de los repositorios que cuentan con todo el trabajo desarrollado en esta actividad:

- API Banco Bogotá: [https://github.com/lfvelascot/api.banco\\_bogota](https://github.com/lfvelascot/api.banco_bogota)

 UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ	Universidad de San Buenaventura Facultad de ingeniería Bases de datos distribuidas	Desarrollo de APIS – Ejercicio del Banco	2021 - 2
---	---	--	----------

- API Banco Medellín: [https://github.com/lvelascot/api.banco\\_medellin](https://github.com/lvelascot/api.banco_medellin)
- API Datacenter: [https://github.com/lvelascot/api.banco\\_datacenter](https://github.com/lvelascot/api.banco_datacenter)

Dentro de los repositorios podrá encontrar la siguiente información:

- Carpeta con el proyecto de la API.
- Carpeta con los scripts con las sentencias DDL y esquema físico de cada base de datos.
- Archivo .json que contiene todas las peticiones de cada API, este archivo lo puede importar a Postman para realizar las pruebas personalmente.

