

# Desarrollo Practica API REST

## Definición del proyecto

El presente proyecto tiene como objetivo desarrollar una API REST , dándole al desarrollador la libertad de elegir el lenguaje de programación y tecnologías usar.

Basado en lo anterior, he elegido lo siguiente para el desarrollo de la práctica:

- Java (EE): El lenguaje de programación de alto nivel Java, debido a que es con el que estoy más familiarizado y tengo mayor dominio. Usare la edición empresarial debido a que contiene API de programación que automatizan en gran medida la creación de configuración de distintos tipos de proyectos web, me permitirá realizar la practica dentro el tiempo establecido.
- JAX-RS :API de java que permite la creación de servicios web con REST .Dispone de anotaciones que sirve para simplificar el desarrollo y despliegue de los servicios clientes y servicios consumidores en la web.
- Jersey: es un framework de Java que implementa JAX-RS, por lo cual mediante el uso de mismo se puede crear rápidamente servicios REST.
- Eclipse (Java EE IDE): Entorno de desarrollo que facilita las tareas de programación y despliegue de aplicaciones.

- Apache Tomcat (7.0): Servidor de aplicaciones que permite desplegar nuestras aplicaciones de forma rápida y sencilla, ahorrándonos la configuración del servidor desde 0.
- Postman: Herramienta ampliamente usada para el envío de peticiones HTTP, que puede ser utilizada como cliente API REST y con la cual haremos las pruebas al proyecto.

A su vez, he definido que la API REST tendrá las siguientes características:

- Definición del negocio: Sera la API REST de un sistema bancario el cual contara para esta practica con la siguiente información:
  - a) Sucursales: Sucursales con las que cuenta el banco para atender a los clientes.
  - b) Clientes : Clientes (cuentahabientes ) del banco que están registrados
- Definición de reglas de negocio:
  - a) Se pueden agregar, consultar o actualizar sucursales.
  - b) Se pueden agregar, consultar, actualizar o eliminar clientes
  - c) Una sucursal puede tener muchos clientes, pero un cliente puede estar registrado solo en una sucursal.

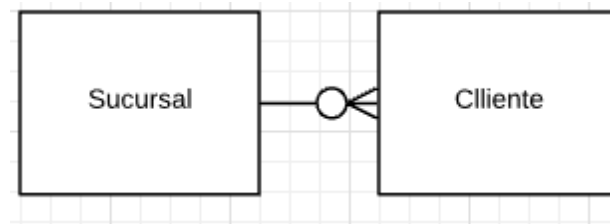


Figura 1. Relación entre la información contenida en la API REST

- Definición de Uniform Resource Identifiers

Tomando en cuenta las reglas del negocio antes definidas, se establecen los URIs de la siguiente forma:

- La colección de recursos de Sucursales bajo la URI:  
*“/api/sps/helloworld/v1/ sucursales”*
- Las instancias de recursos de Sucursales bajo URIs:  
*“/api/sps/helloworld/v1/ sucursales/{sucursalId}”*
- La colección de recursos de Clientes de una sucursal bajo la URI:  
*“/api/sps/helloworld/v1/ sucursales/{sucursalId}/clientes”*
- Las instancias de recursos de Clientes de una Sucursal bajo URIs :  
*“/api/sps/helloworld/v1/  
sucursales/{sucursalId}/clientes/{clienteId}”*

- Definición de mensajes:

Todas las solicitudes se harán mediante peticiones http usando los verbos propios de REST ( GET,POST,PUT, DELETE) Mientras que las respuestas darán como resultado mensajes JSON dentro del body del mismo. Se definen los mensajes para nuestros mensajes de la siguiente forma:

### 1. Mensaje Sucursal solicitud POST,PUT

```
{  
  "nombre": "Vallejo",  
  "direccion": "Sobre vallejo ",  
  "actEstado": "2020-02-20T01:27:36.000",  
  "estado": "En operacion"  
}
```

Donde el campo nombre es el nombre de la ubicación de la sucursal, dirección es la dirección completa, actEstado es la fecha en la que se actualizo el estado del banco y estado indica la condición de la sucursal como pueden ser “En operacion” , “En Remodelacion ” ó “Cerrado”

### 2. Mensaje Sucursal respuesta POST,GET(Instancia),PUT

```
1  
2  "actEstado": "2020-02-20T01:27:36Z[UTC]",  
3  "clientes": {},  
4  "direccion": "Calle de Vallejo #215",  
5  "estado": "En operacion",  
6  "id": 1,  
7  "nombre": "Vallejo Banco"  
8 }
```

Donde se agrega el campo id, indicando el identificador único de esa sucursal, y el campo clientes, que es una colección de los clientes registrados en esta sucursal

### 3. Mensaje Sucursal respuesta GET(Colección)

```
1  {
2    {
3      "actEstado": "2020-02-20T01:27:36Z[UTC]",
4      "clientes": {},
5      "direccion": "Calle de Vallejo #215",
6      "estado": "En operacion",
7      "id": 1,
8      "nombre": "Vallejo Banco"
9    },
10   {
11     "actEstado": "2018-02-20T12:40:00Z[UTC]",
12     "clientes": {},
13     "direccion": "Calle de Lindavista #22",
14     "estado": "En operacion",
15     "id": 2,
16     "nombre": "Lindavista MiBanco"
17   },
18   {
19     "actEstado": "2019-02-20T03:40:00Z[UTC]",
20     "clientes": {},
21     "direccion": "Calle de Lindavista #22",
22     "estado": "Cerrado",
23     "id": 3,
24     "nombre": "Iztapalapa MiBanco"
25   }
26 }
```

Se obtiene la colección de las sucursales registradas en ese momento de hacer la petición.

### 4. Mensaje Cliente de una sucursal solicitud POST,PUT

```
1  {
2    "nombre" : "Daniel Isaías Herrera Monter",
3    "fecharegistro" : "2019-02-20T03:40:00.000",
4    "ocupacion" : "Estudiante",
5    "ingresomensual": "0"
6  }
```

Donde se tiene nombre del cliente, la fecha de registro indicando la fecha en que se dio de alta en el sistema, Su ocupación y su ingreso mensual.

## 5. Mensaje Cliente de una sucursal respuesta POST,PUT,GET (Instancia)

```
1 {  
2   "fecharegistro": "2019-02-20T03:40:00Z[UTC]",  
3   "id": 1,  
4   "ingresomensual": "0",  
5   "nombre": "Daniel Isaías Herrera Monter",  
6   "ocupacion": "Estudiante"  
7 }
```

Donde se agrega el campo de Id , indicando el identificador del cliente en la sucursal.

## 6. Mensaje Cliente de una sucursal respuesta GET (Coleccion)

```
1 [  
2   {  
3     "fecharegistro": "2019-02-20T03:40:00Z[UTC]",  
4     "id": 1,  
5     "ingresomensual": "0",  
6     "nombre": "Daniel Isaías Herrera Monter",  
7     "ocupacion": "Estudiante"  
8   },  
9   {  
10    "fecharegistro": "2013-01-24T13:40:00Z[UTC]",  
11    "id": 2,  
12    "ingresomensual": "2000",  
13    "nombre": "Miriam Monter Torres",  
14    "ocupacion": "Ama de Casa"  
15  }  
16 ]
```

Se obtiene la colección de las sucursales registradas en ese momento de hacer la petición.

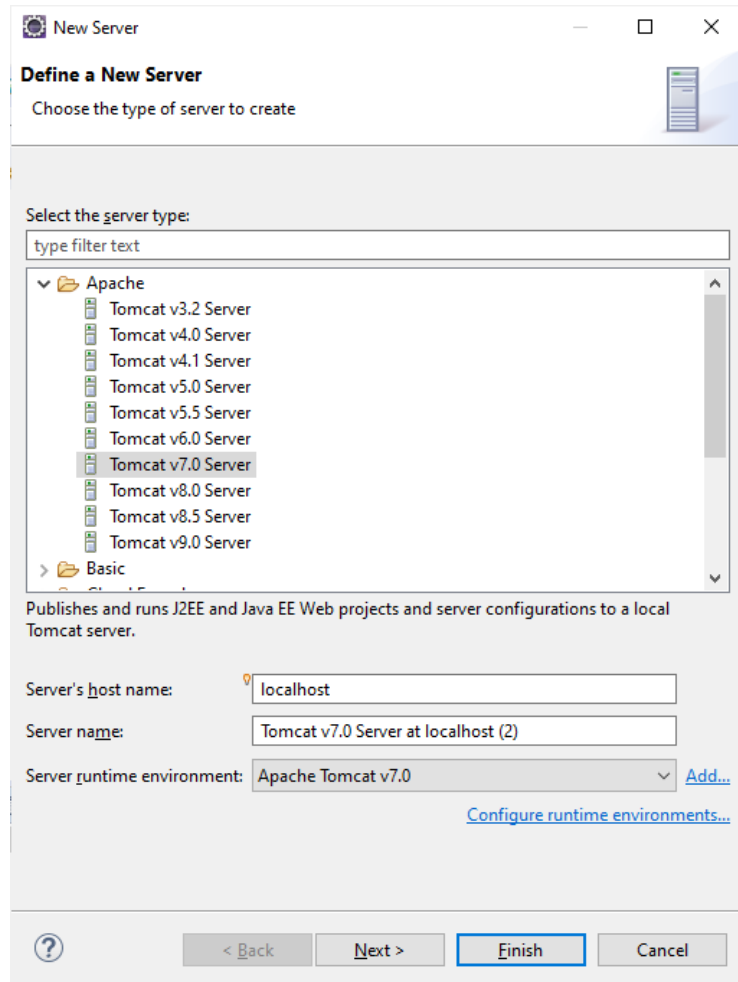
## Desarrollo del proyecto.

Ya que tenemos bien definido nuestra API REST, pasamos a la parte de codificación e implementación.

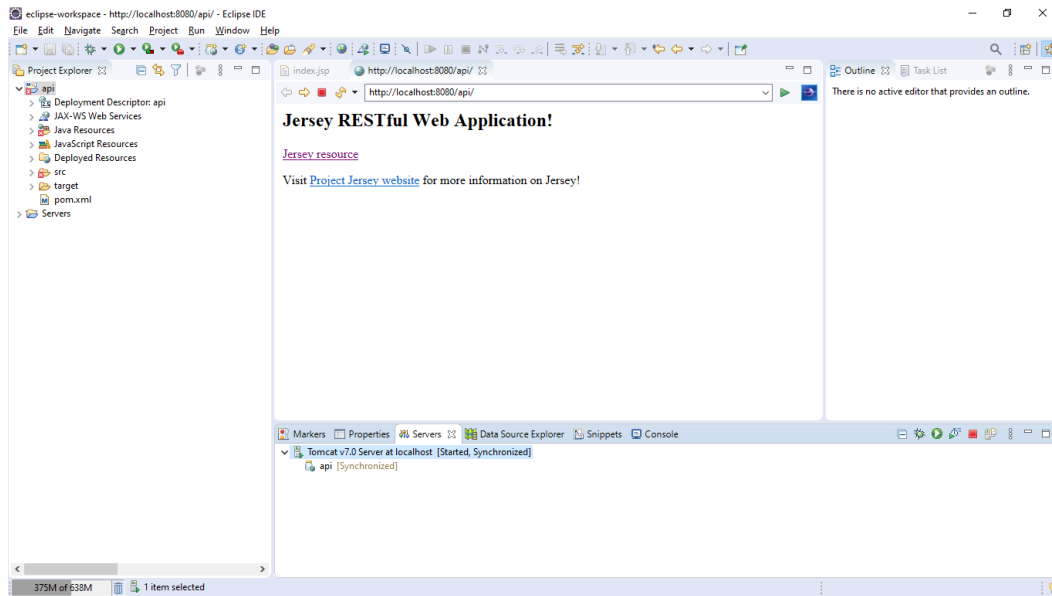
1. Abrir el IDE y comenzar un proyecto, en este caso mediante el gestor de proyectos maven podemos seleccionar un arquetipo

que nos brindara la estructura necesaria para este tipo de proyectos.

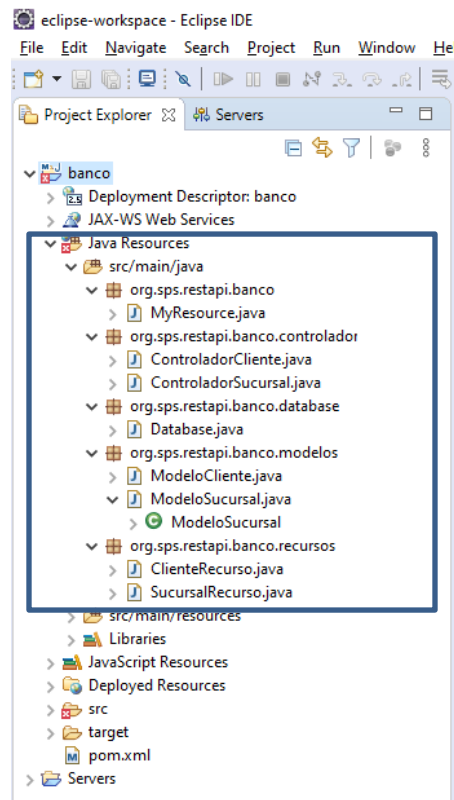
2. Elegimos un servidor para alojar nuestra aplicación, en mi caso Tomcat 7.0 de apache pues ya he trabajado con el. Posterior agregamos nuestra aplicación.



### 3. Hasta aquí ya tenemos la estructura y configuración del servidor.



### 4. Ahora paso a realizar la implementación de los recursos y funcionalidad. Quedando la estructura final del proyecto de la siguiente forma.





Como tal la implementación del proyecto se encuentra en el directorio del recuadro, todos los archivos fuentes generados están aquí bajo el paquete raíz `org.sps.restapi.banco`

Aquellos archivos bajo el subpaquete `.controlador` son los archivos fuentes que permiten al momento de desplegar el servidor web, resolver las distintas peticiones a los recursos de la API Rest. Estos hacen uso del subpaquete `.recursos` pues es aquí donde se realizan las reglas de negocio de la API REST

Aquellos archivos bajo el subpaquete `.database` son los archivos fuentes que trabajan como la clase de base de datos, es importante mencionar que debido al tiempo como tal se implementó algún tipo de base de datos o persistencia. Sin embargo bajo este modelo es posible en un futuro realizar una implementación y solo se cambiaría la implementación de esta clase. Aun así, mientras la aplicación esta corriendo presenta las características del CRUD que una API REST debe tener.

Aquellos archivos bajo el subpaquete `modelos` son las clases que contiene la definición de los mensajes con los que trabaja la API.

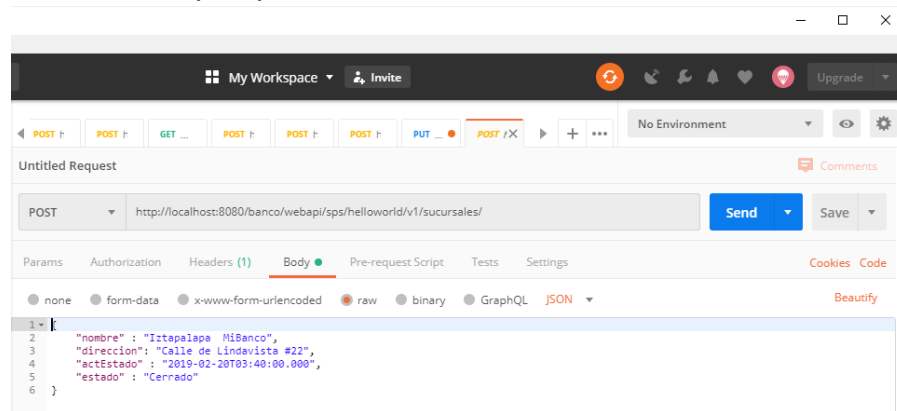
Por último el subpaquete de recursos es el encargado de realizar las reglas de negocio para cada tipo de mensaje.

## Pruebas de Funcionamiento

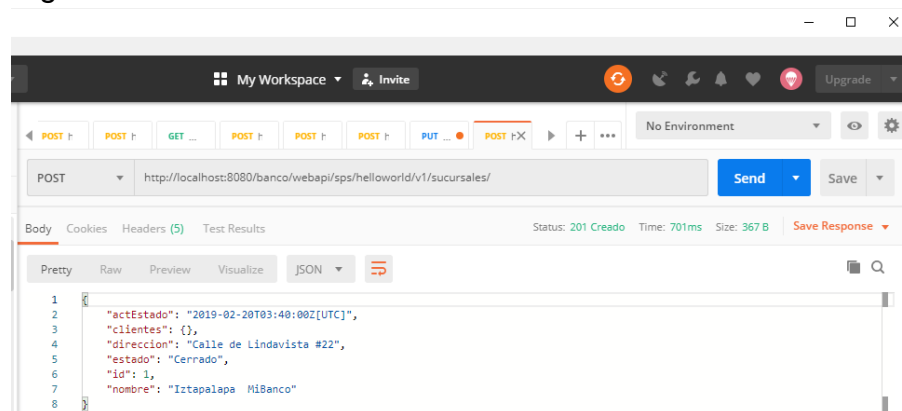
- Registro de Sucursales

Para llevar a cabo el registro de sucursales tenemos que hacer lo siguiente. Con cualquier cliente API REST realizar una petición http de tipo POST a la URI “<http://localhost:8080/banco/webapi/sps/helloworld/v1/sucursales/>”, con header “Content - Type : application/json” y en el Body seleccionar el formato “raw” y crear un mensaje con el formato especificado en la sección *Definición del proyecto*.

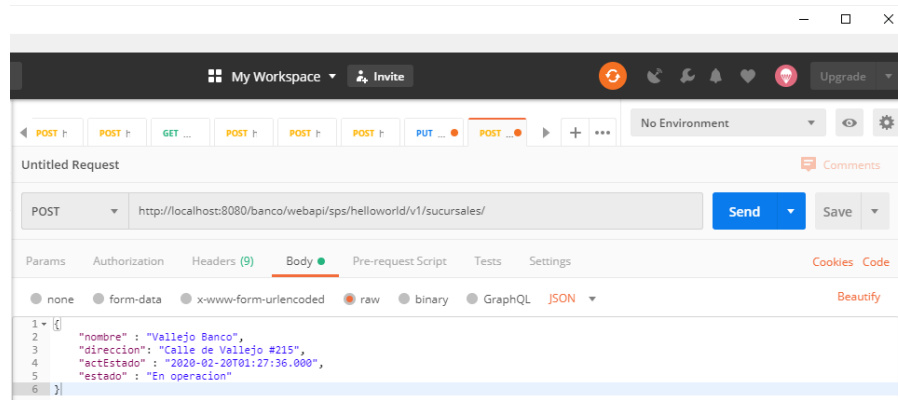
En mi caso la realizare con POSTMAN y damos al botón send.  
Sucursal Iztapalapa



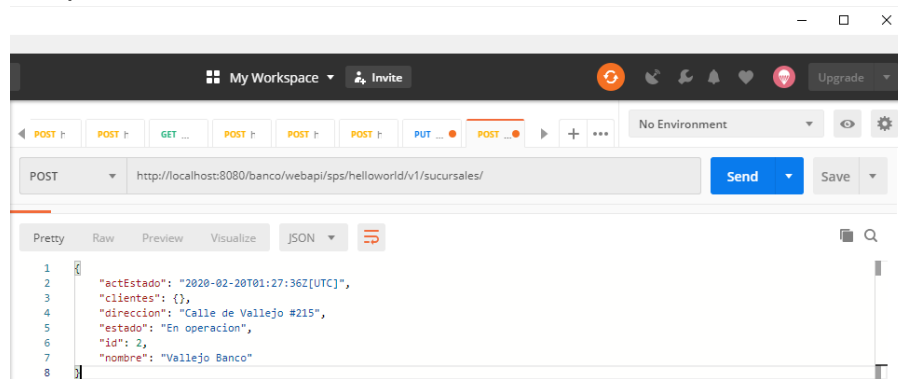
Mandara la siguiente respuesta, indicando la creación exitosa del registro.



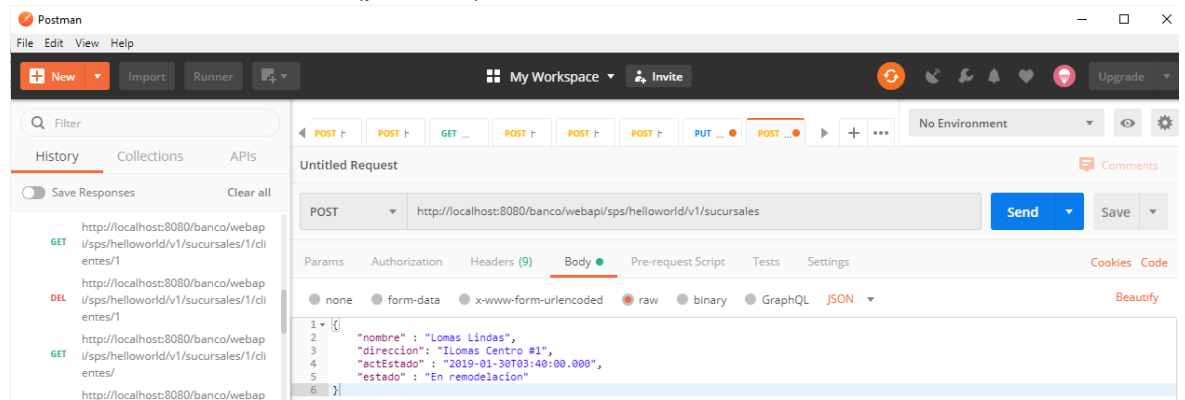
Hacemos lo mismo con otras dos sucursales:  
-Sucursal Vallejo (peticion)



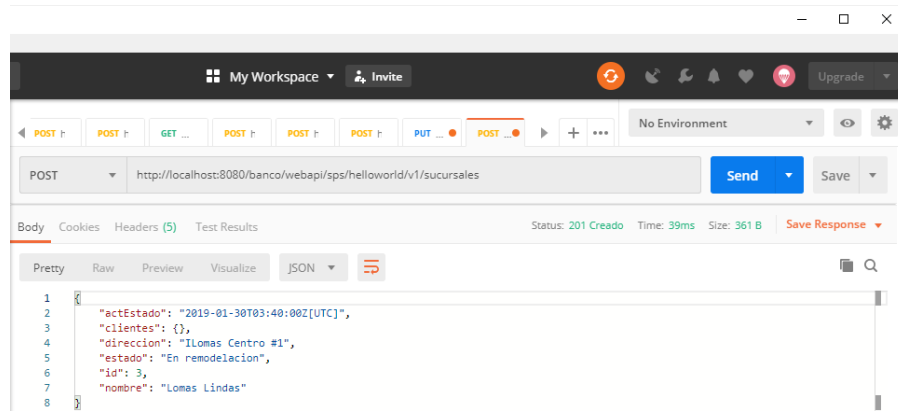
## Respuesta



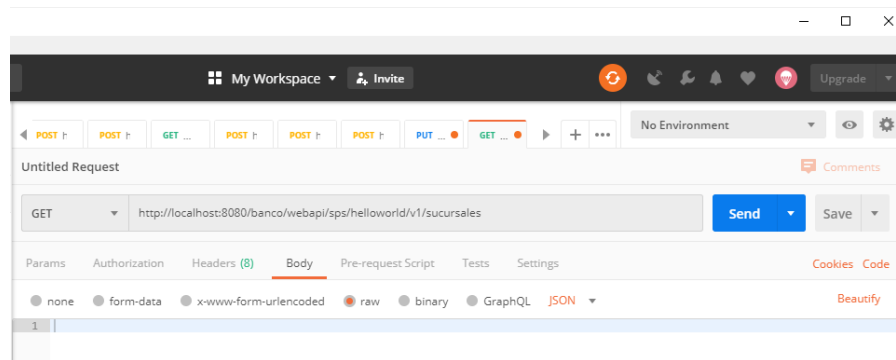
## -Sucursal LomasLindas (peticion)



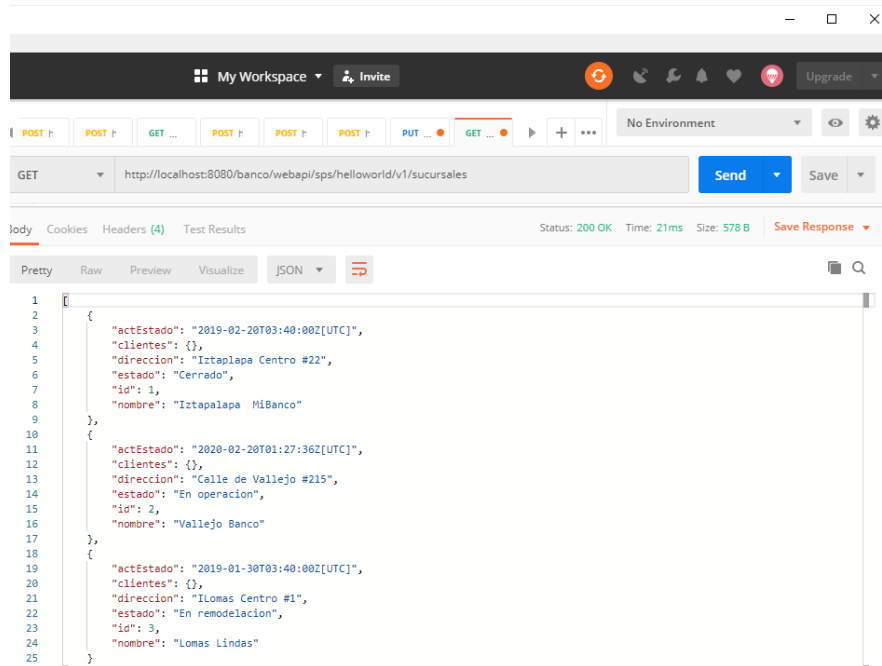
## Respuesta



- **Consulta de Sucursales (Colección)**  
Para llevar a cabo la consulta de todas las sucursales tenemos registradas que hacer lo siguiente. Con cualquier cliente API REST realizar una petición http de tipo GET a la URI  
“http://localhost:8080/banco/webapi/sps/helloworld/v1/sucursales”.



Respuesta



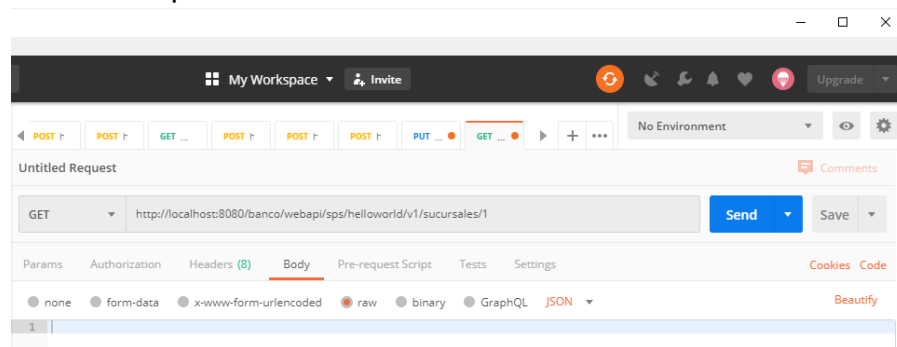
- **Consulta de Sucursales (Instancias)**

Para llevar a cabo la consulta de una sucursal que este registrada se debe de conocer su id. Con cualquier cliente API REST realizar una peticion http de tipo GET a la URI

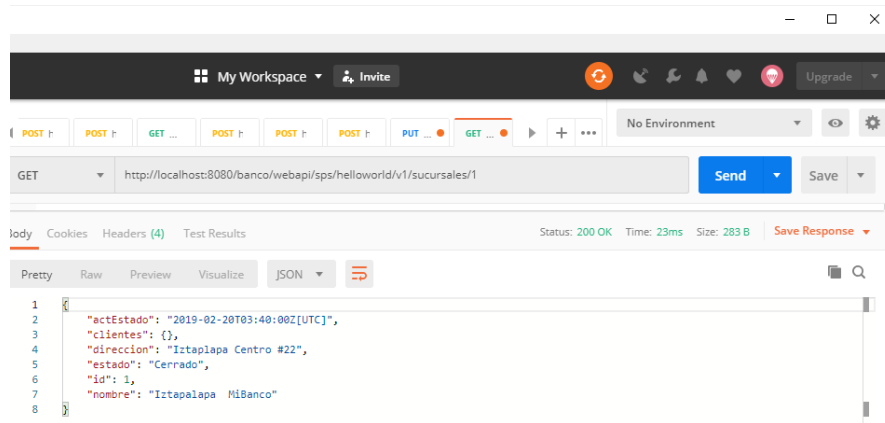
"http://localhost:8080/banco/webapi/sps/helloworld/v1/sucursales/{sucursalId}".

Donde {sucursalId} se debe de sustituir por el id de la sucursal a consultar. Ejemplo para conocer la sucursal 1 y 3

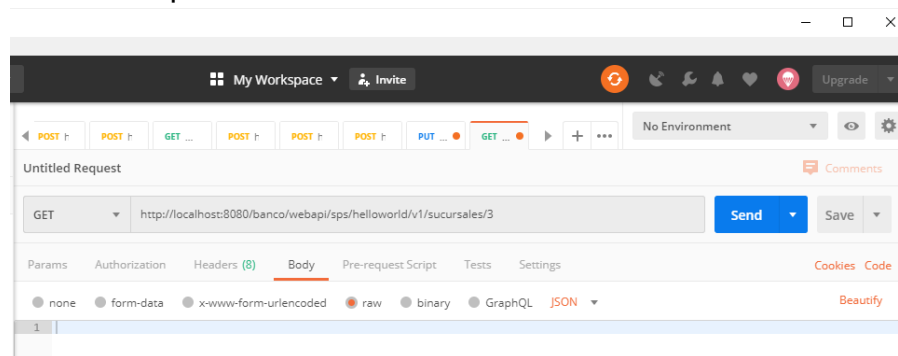
### Sucursal 1 peticion



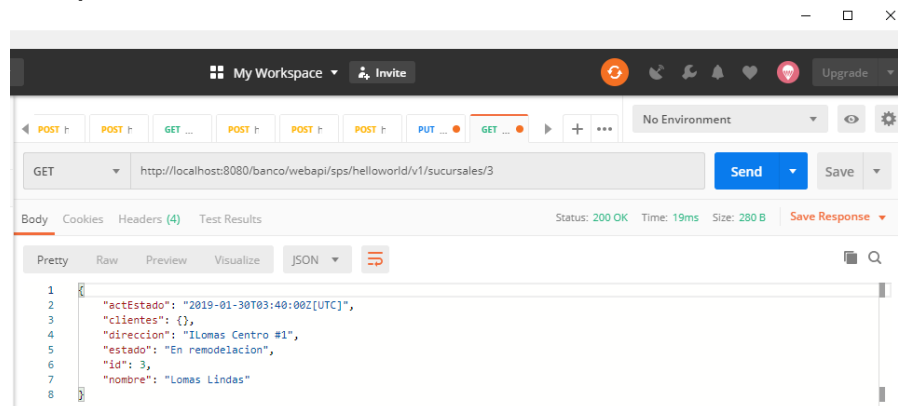
### Respuesta



## Sucursal 3 peticion



## Respuesta



- Actualización de Sucursal

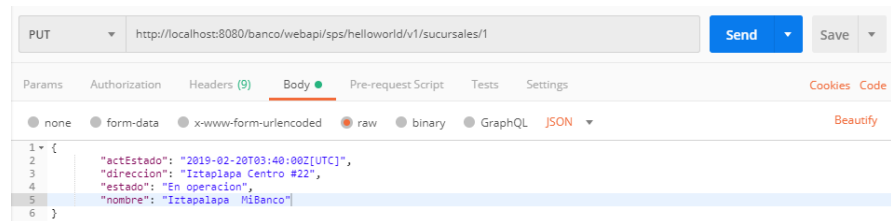
Para llevar a cabo la actualización de sucursales tenemos que hacer lo siguiente. Con cualquier cliente API REST realizar una petición http de tipo PUT a la URI

“<http://localhost:8080/banco/webapi/sps/helloworld/v1/sucursales/{sucursalId}>“, Donde {sucursalId} se debe de sustituir por el id de la sucursal a actualizar, además debe ir con el header “Content - Type : application/json” y en el

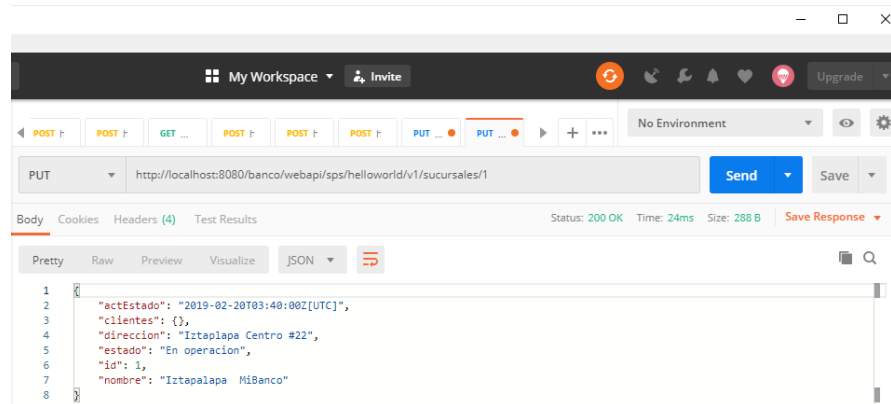
Body seleccionar el formato “raw” y crear un mensaje con el formato especificado en la sección *Definicion del proyecto* que incluya todos los campos con la información reciente.

En este caso actualizaremos la sucursal con id 1, que es la de Iztapalapa e indicaremos que está nuevamente en operación

### Solicitud

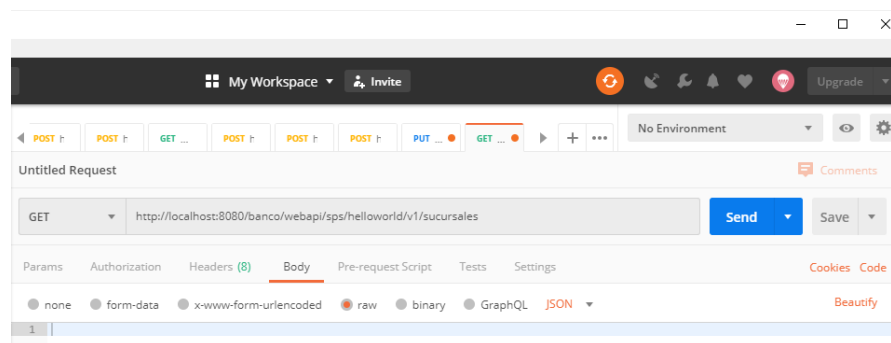


### Respuesta

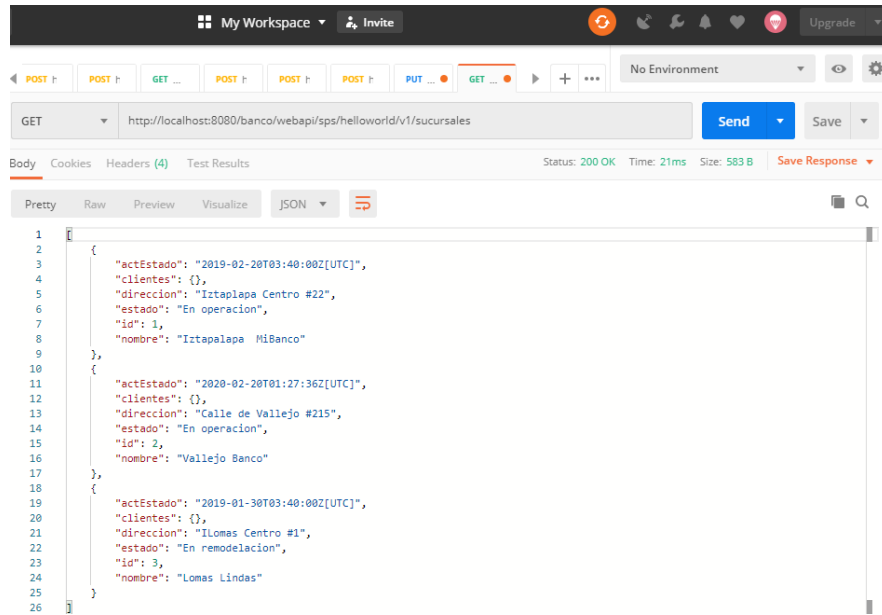


Para confirmar que se realizó correctamente la actualización, volvemos a consultar todas las sucursales y vemos reflejado el cambio.

### Solicitud



### Respuesta



```
1 {
2   {
3     "actEstado": "2019-02-20T03:40:00Z[UTC]",
4     "clientes": {},
5     "direccion": "Iztapalapa Centro #22",
6     "estado": "En operacion",
7     "id": 1,
8     "nombre": "Iztapalapa MiBanco"
9   },
10  {
11    "actEstado": "2020-02-20T01:27:36Z[UTC]",
12    "clientes": {},
13    "direccion": "Calle de Vallejo #215",
14    "estado": "En operacion",
15    "id": 2,
16    "nombre": "Vallejo Banco"
17  },
18  {
19    "actEstado": "2019-01-30T03:40:00Z[UTC]",
20    "clientes": {},
21    "direccion": "I Lomas Centro #1",
22    "estado": "En remodelacion",
23    "id": 3,
24    "nombre": "Lomas Lindas"
25  }
26 }
```

- Registro de Clientes
- Consulta de Clientes (Coleccion)
- Consulta de Clientes (Instancia)
- Actualización de Clientes
- Eliminar Clientes



Daniel Isaías Herrera Monter  
19/02/2020