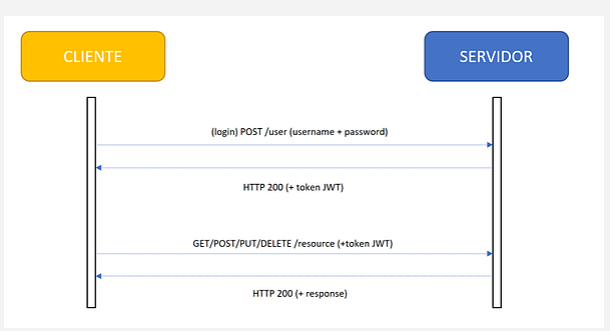
**Crear un Proyecto con Spring Boot**

1. Crear una implementación de logueo con JWT. - Se definirá mediante un usuario y contraseña que lo obtendrá desde un archivo de texto (debe tener una lista de usuarios). Luego, si es correcto podrá generar un token de acceso para acceder a los recursos protegidos del carrito de compras indicados en el punto 2.



Tanto para éxito o error deberá guardar en un solo archivo log (con log4j2) los siguientes campos:

* Fecha y Hora
* Usuario
* Tipo de Mensaje:
  + Si es error: Guardar mensaje ‘*error’* y de tipo “**error**”.
  + Si es éxito: Guardar mensaje ‘*Logueo exitoso* y de tipo “**debug**”.
  + Si el usuario y contraseña son incorrectas: Guardar mensaje ‘*datos incorrectos en el logueo’* y de tipo “**info**”.

1. En el carrito de compras deberá tener en cuenta las siguientes implementaciones:

* Cliente **[R]**
* Producto **[G]**
* Venta **[RC]**

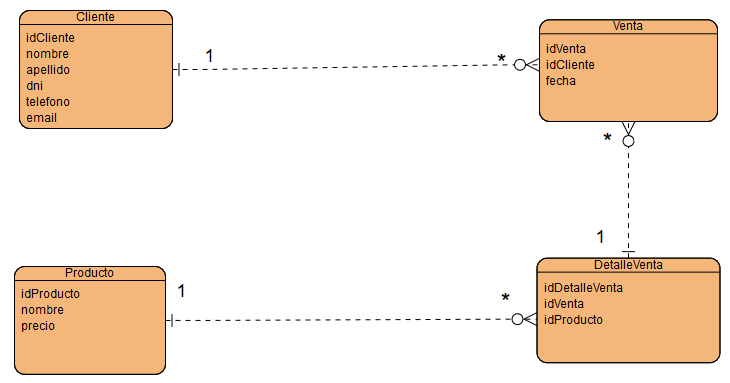
Leyenda:

**[G]** = Gestión Completa (CRUD)

**[R]** = Registro

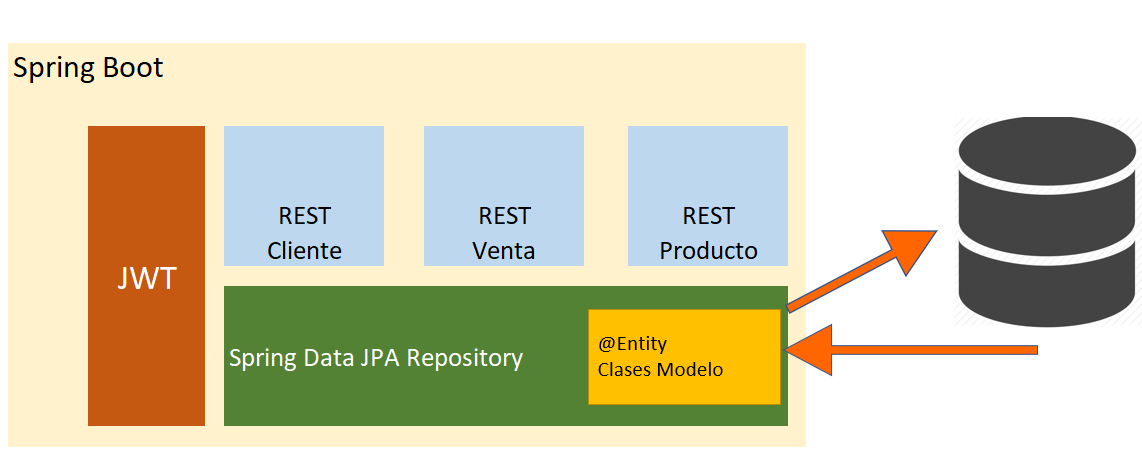
**[RC]** = Registro y Consulta

**Modelo de Base de datos:**



El modelo de datos se debe manejar en cascada en el uso de los servicios Rest.

**Perspectiva Back-end (Monolítico):**



Debe tener las siguientes consideraciones:

* La seguridad de los servicios será mediante JWT.
* Los servicios REST solo deben manejar JSON como notación.
* Hacer uso de pool de conexiones.
* Hacer uso de los componentes de Spring Boot (Controller, Service, Persistence).
* Hacer uso de las anotaciones en todos los componentes.
* La configuración de Base de Datos y JWT debe ser configurable.
* Crear un manejador de errores para el carrito de compras que permita capturar los errores que se generen de tal forma que devuelva un JSON estandarizado, lo cual tendrá los siguientes atributos:
  + **HttpStatus**: Tendrá el status http que devolverá la petición REST.
  + **Message**: Sera usado para mostrar un pequeño mensaje de error que describa el problema.
  + **Code**: Define algún código de error que pueda ser representativo para una aplicación.
  + **BackendMessage**: Se entiende como la excepción generada para que un desarrollador pueda detectar de forma simple el problema.

\* Evidenciar un caso de error con cualquier servicio Rest y subirlo a GitHub.

* Hacer uso del Log4j2 para errores. - Cada servicio Rest deberá tener su propio archivo log con la siguiente estructura:
* Fecha y Hora.
* Los atributos definidos en el manejador de errores.
* Implemente un nuevo servicio Rest que haga la búsqueda por “idVenta” o “idCliente” usando RxJava con expresiones lambdas para obtener la lista del detalle de la venta.
* Utilice alguna librería de Spring Boot para realizar pruebas unitarias.
  + Crear y ejecutar un caso de prueba de éxito por cada Servicio Rest del CRUD Cliente. (guardar evidencia y subir a GitHub)
  + Crear y ejecutar un caso de prueba de error para la entidad Producto. (guardar evidencia y subir a GitHub)
  + Crear y ejecutar un caso de prueba de error para la entidad Venta. (guardar evidencia y subir a GitHub)

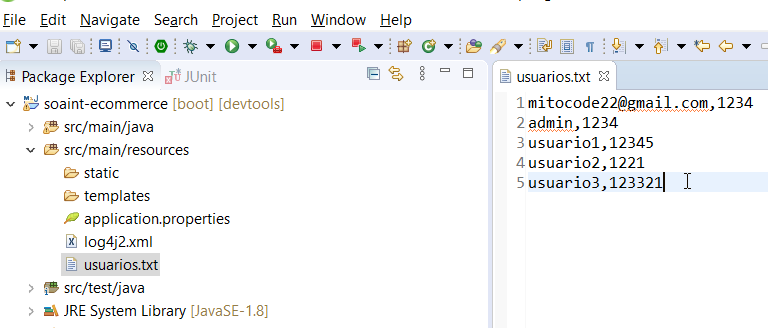
1. Organizar la solución en contenedores con Docker.
2. Con sus propias palabras indique de qué forma podría mejorar la seguridad y optimizar el carrito de compras.

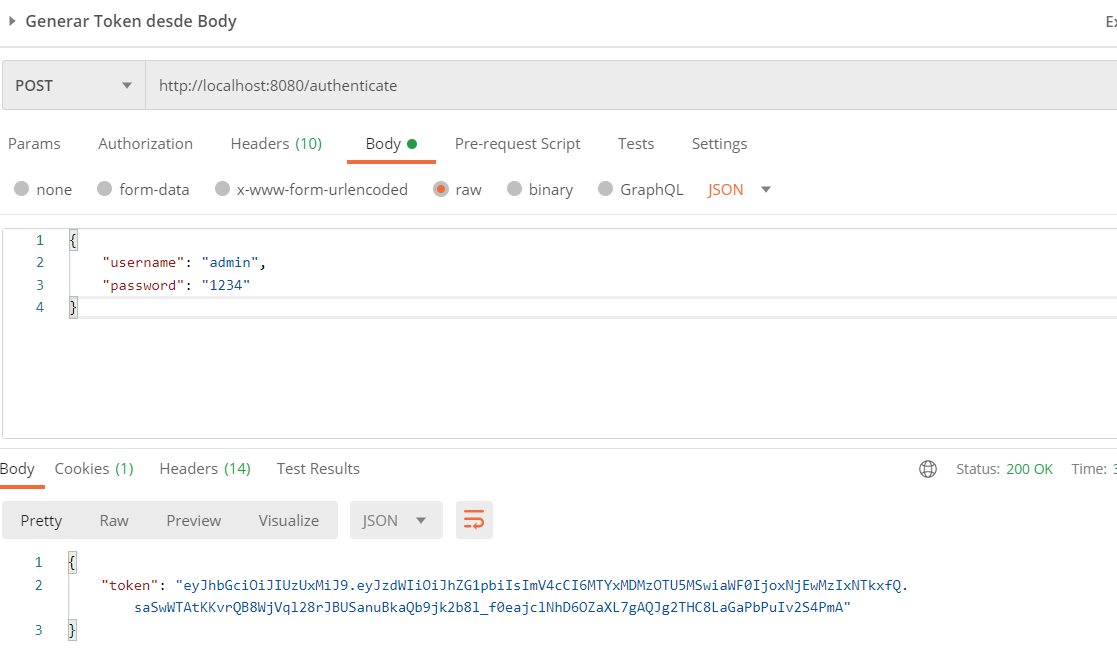
Se podria mejorar la seguridad con la implementación de oauth2 y ademas encriptando la información los datos de entrada y salida.

Para optimizar se podria llevar el proyecto a un arquitectura de microservicios.

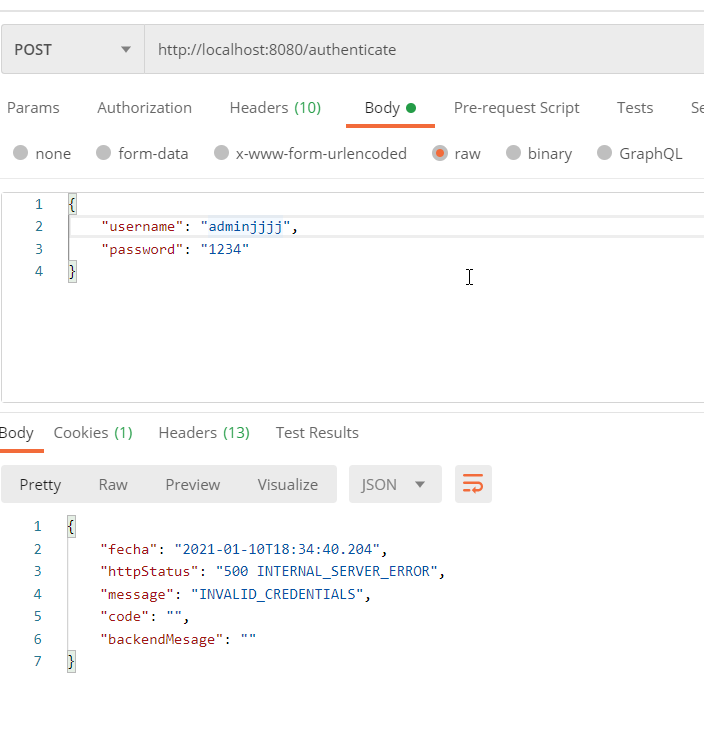
1. Finalmente la solución debe estar publicada en su cuenta de Github.

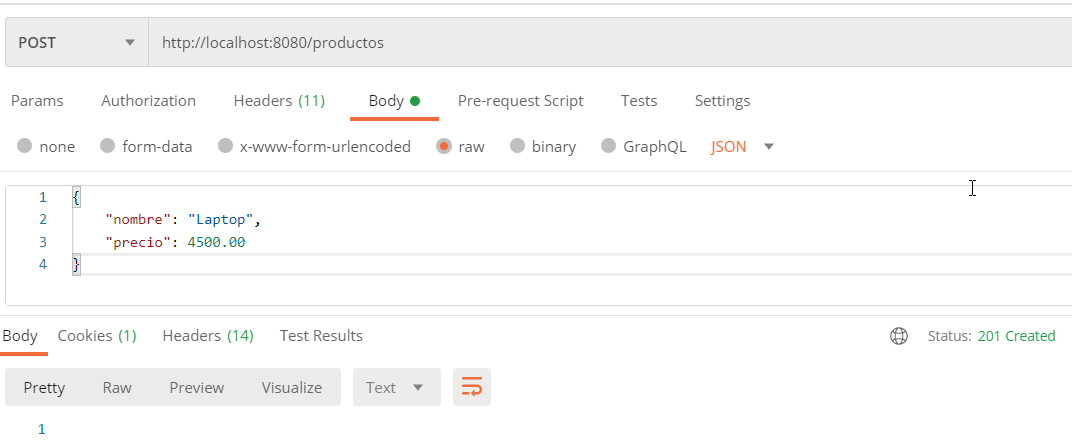
Evidencias:

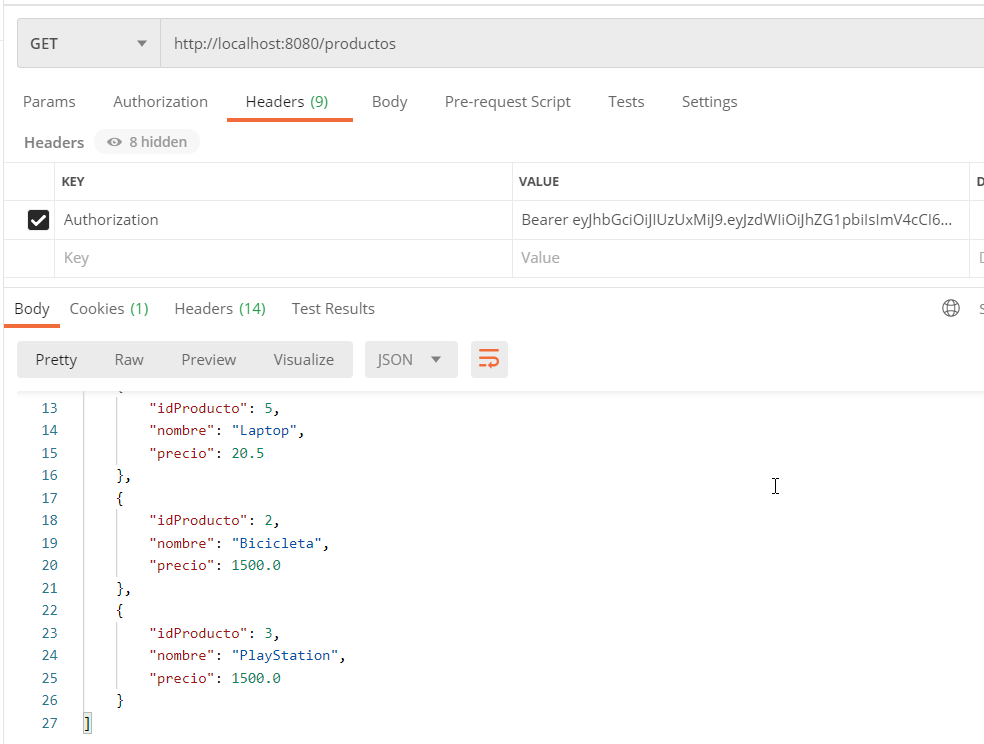
* + - Authenticacion

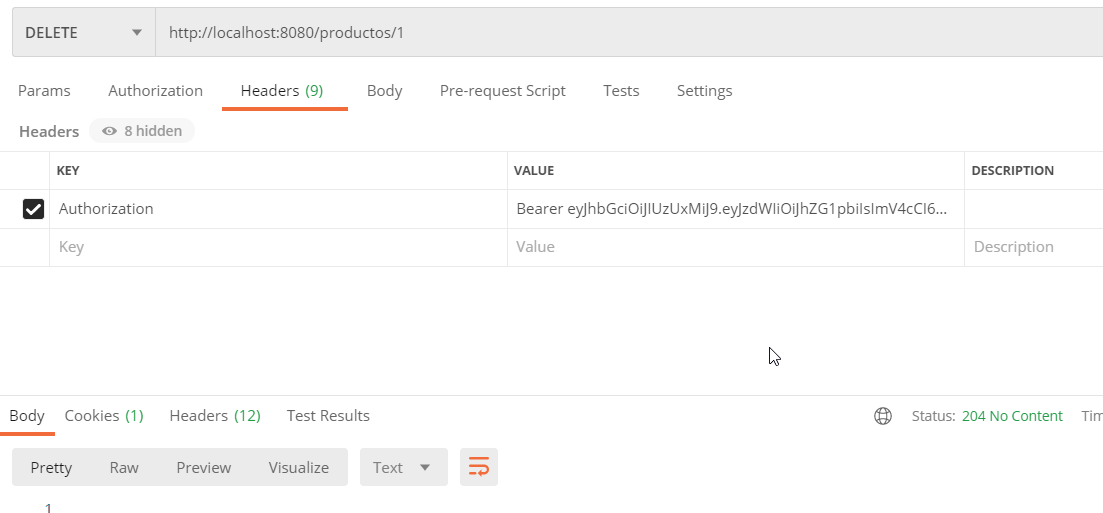


* + - Caso Error :

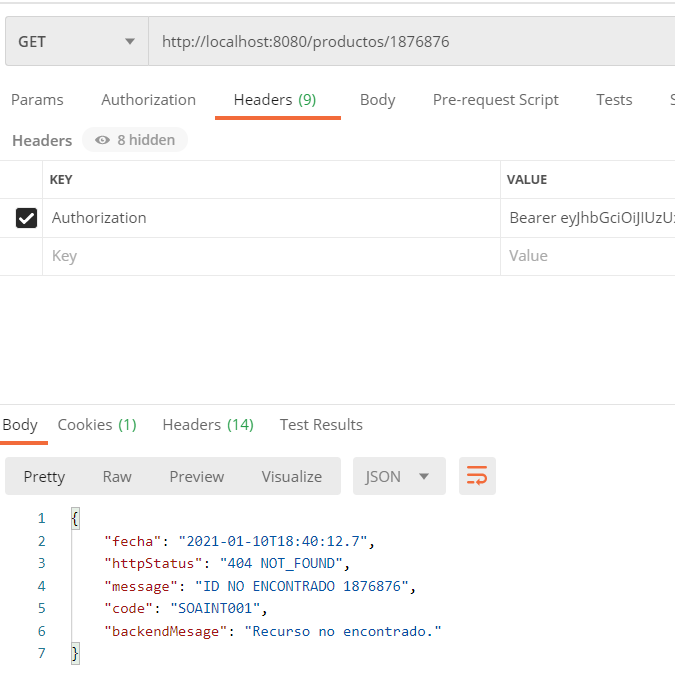


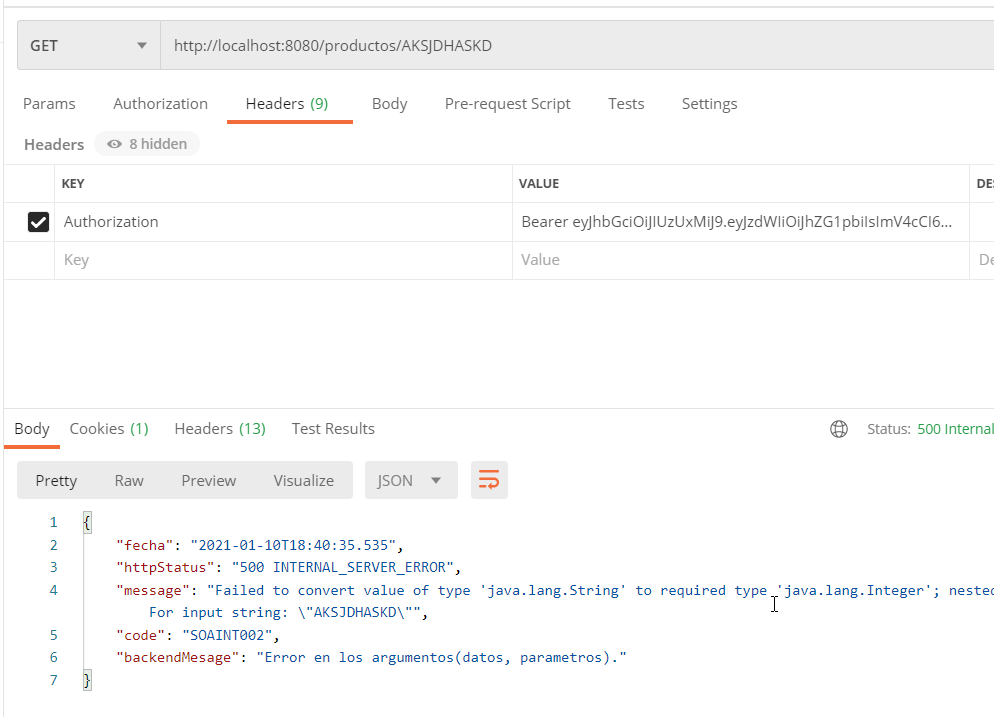
* + - Servicio - Producto



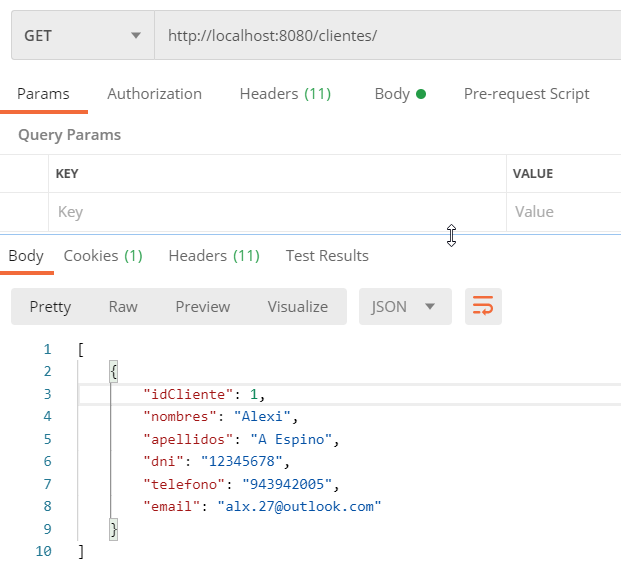


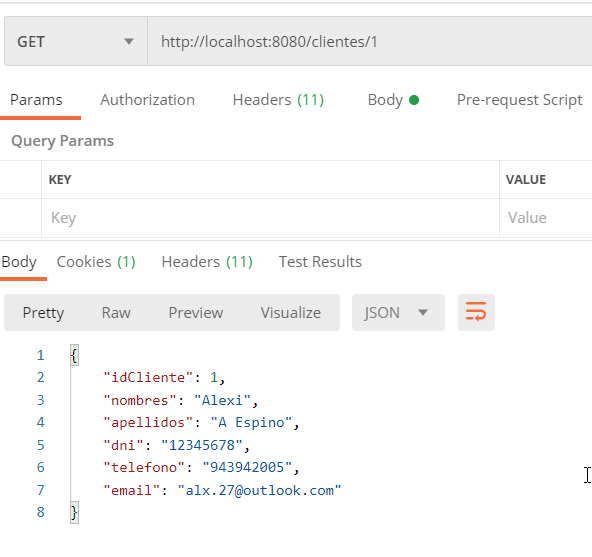
* + - Caso Error :



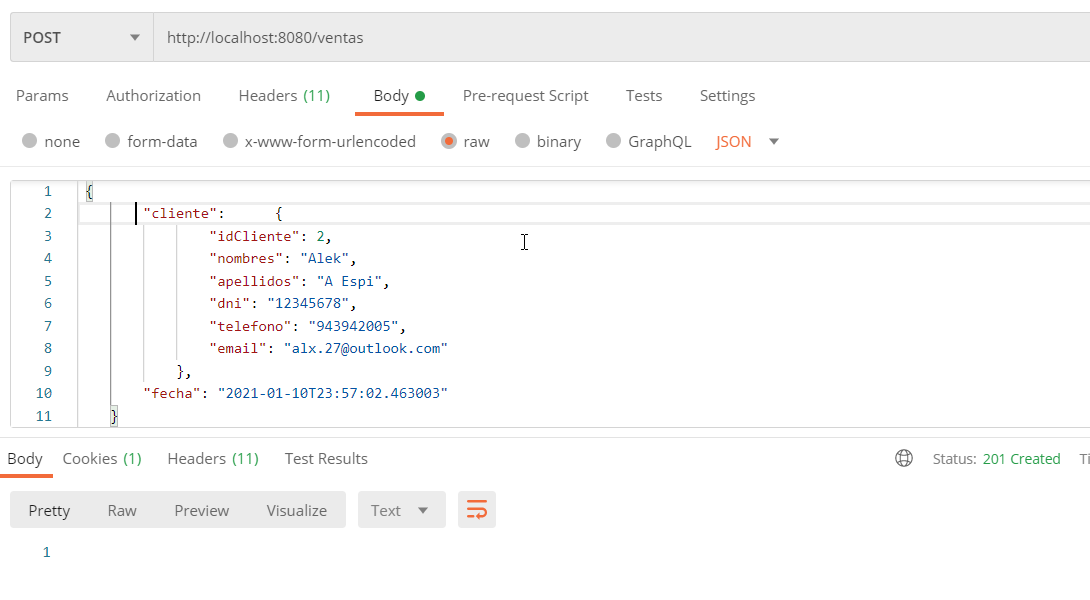


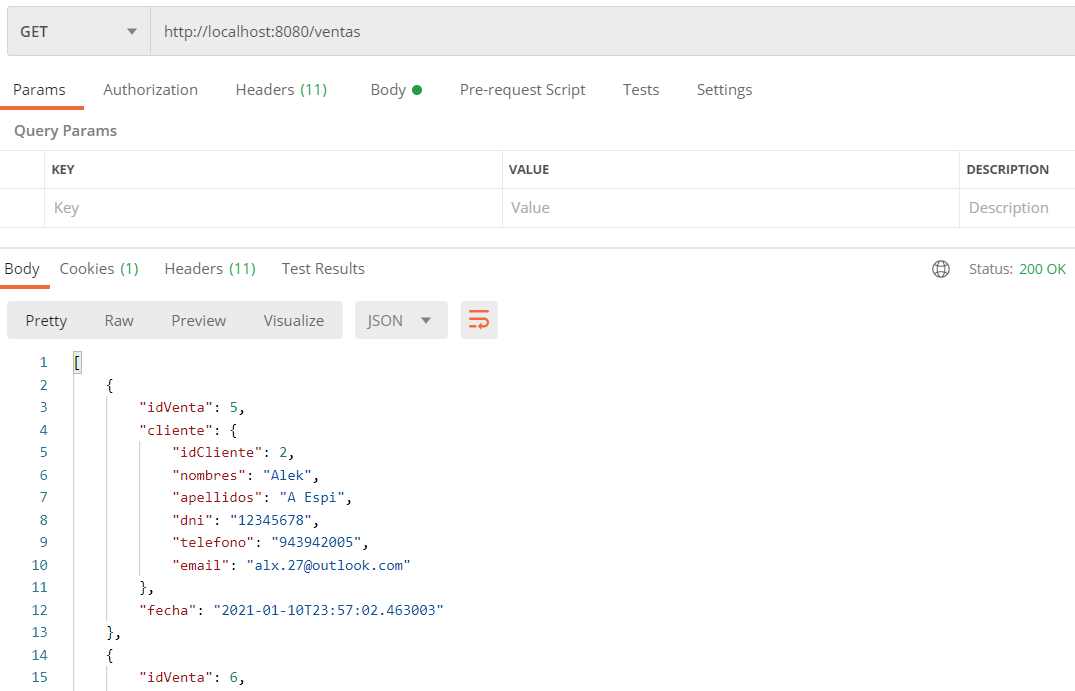
* + - Servicio Cliente





* + - Servicio Venta





Pruebas Unitarias:

