Aplicaciones con Enfoque Orientado a Servicios

Tema Nº3:Servicios web REST

Indicador de logro Nº3:Construye servicios web utilizando el estilo de arquitectura REST

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº3:**

Servicios web REST

**Subtema 3.1:**

Conceptos básicos y buenas prácticas

El objetivo principal de cualquier sistema distribuido es facilitar el acceso a recursos remotos. *Representational State Transfer* o REST es un estilo de arquitectura utilizado en el desarrollo de servicios web, muy diferente a aquellos que se realizan con el protocolo SOAP. REST se diseñó pensando en ser simple para lograr una rápida adopción del usuario y un desarrollo rápido.

Los servicios que implementan REST tienen las siguientes características:

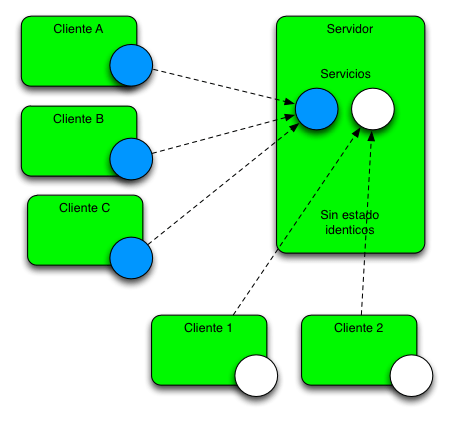
* Cliente / Servidor

Se define una interface de comunicación entre ambos separando completamente las responsabilidades de cada parte.



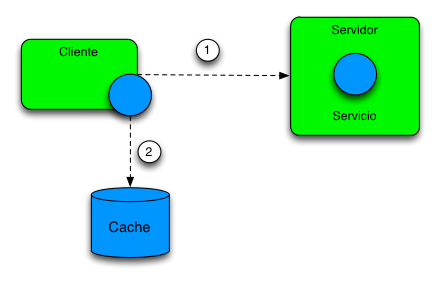
* Sin estado

Los servicios web REST no mantienen ningún estado asociado al cliente. Cada petición que se realiza es completamente independiente de la siguiente.



* Caché

El contenido se puede almacenar de tal forma que, una vez realizada la primera petición al servicio, las siguientes peticiones pueden apoyarse en la caché si fuera necesario.

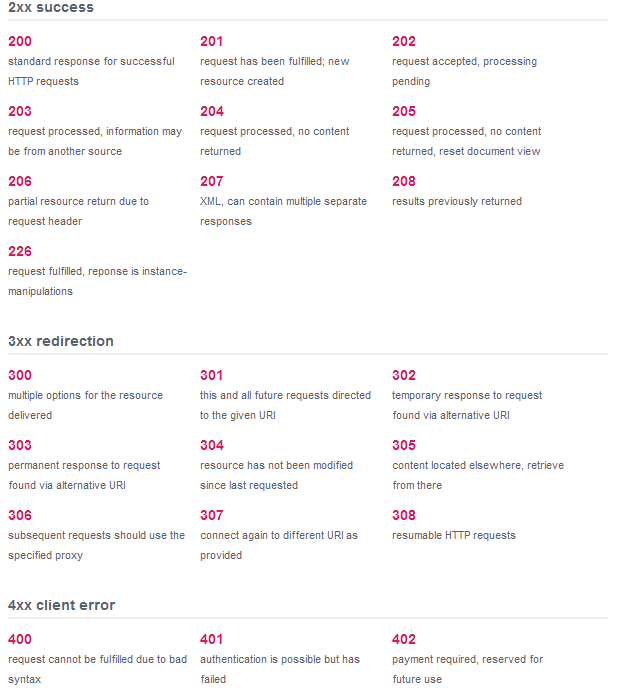


Los servicios web basados en la arquitectura REST, también llamados servicios web RESTful, se basan en **recursos**. Un recurso es una entidad, la cual se almacena principalmente en un servidor y el cliente solicita el recurso utilizando servicios Web RESTful.

A continuación se sugieren buenas prácticas a considerar al momento de desarrollar servicios REST:

* Al devolver una respuesta desde el servidor, agregar el código de estado HTTP correspondiente. Estas respuestas se agrupan en cinco clases:
  + Respuestas informativas (100–199)
  + Respuestas satisfactorias (200–299)
  + Redirecciones (300–399)
  + Errores de los clientes (400–499)
  + errores de los servidores (500–599)
* Utilizar los métodos HTTP según la operación a realizar, no únicamente GET o POST.
* Manejar control de versiones para los servicios para permitir compatibilidad con los clientes.
* Incluir un token de sesión en encabezados en el caso de solicitudes privadas.
* En el caso de resultados extensos, use paginación.
* Definir el servicio para devolver lo que se está solicitando.
* Devolver los tipos de datos correctos, no solo cadenas.

**Ejemplos:**



Códigos de estado HTTP

**Subtema 3.2:**

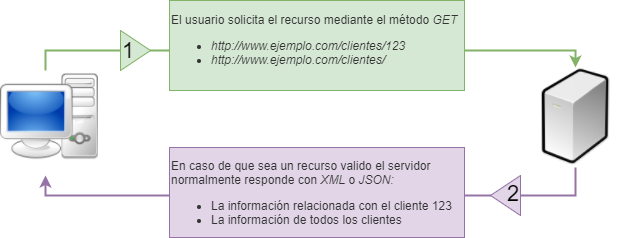
Operaciones básicas

Para acceder o manipular los recursos del servidor, los servicios web que implementan REST deben utilizar los métodos disponibles en HTTP. Los principales según la operación a realizar son:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operación | Método HTTP | URI | Parámetros | Resultado |
| Listar | GET | /{recurso} | - | Lista del tipo del recurso |
| Crear | POST | /{recurso} | Dentro del cuerpo en el POST | Se crea un nuevo recurso |
| Leer | GET | /{recurso}/{id} | - | Un recurso en función del id |
| Actualizar | PUT | /{recurso}/{id} | Dentro del cuerpo en el PUT | Se actualiza el recurso |
| Borrar | DELETE | /{recurso}/{id} | - | Se elimina el recurso en función del id |

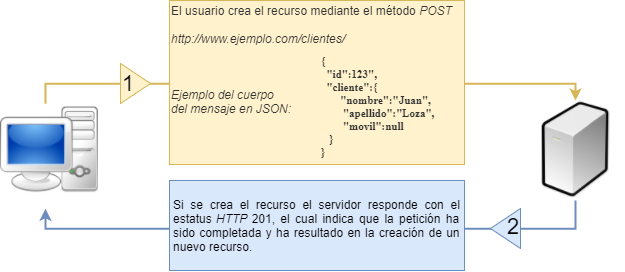
**Ejemplos:**

**GET**



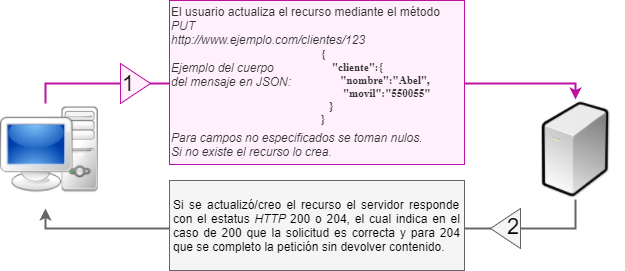
Fuente: <https://blog.bi-geek.com>

**POST**



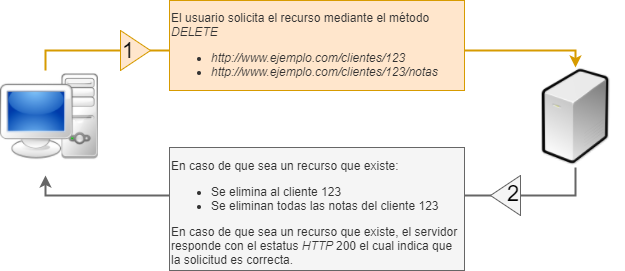
Fuente: <https://blog.bi-geek.com>

**PUT**



Fuente: <https://blog.bi-geek.com>

**DELETE**



Fuente: <https://blog.bi-geek.com>

**Subtema 3.3:**

Formatos básicos de intercambio de información

El formato recomendado para un servicio web RESTful es JSON, sin embargo, también puede retornar la respuesta en formato XML, texto plano, HTML o en secuencia de bytes.

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos. JSON es de fácil lectura y escritura para los usuarios. Por estas características, es el formato predilecto en REST.

**Ejemplos:**

El contenido de JSON va entre llaves, y para arreglos o lista de objetos dentro de corchetes. Además, JSON se caracteriza porque se basa en una estructura clave o identificador – valor, sin necesidad de especificar el tipo de dato que contiene el valor.



**Subtema 3.4:**

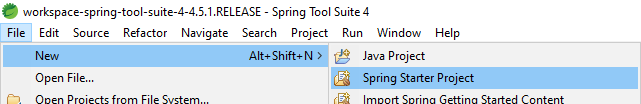
Creación y consumo de servicios web REST

Para crear un servicio web del tipo RESTful utilizaremos la herramienta Spring Tool Suite, Maven y Java 8.

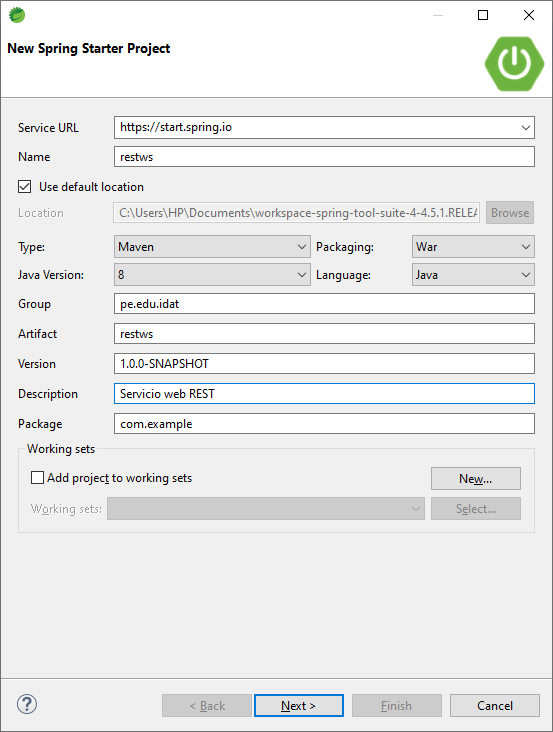
**Ejemplos:**

**Creación de servicio web SOAP**

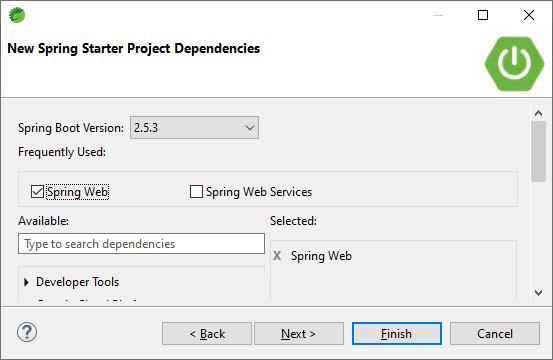
1. En Spring Tool Suite vaya al menú *File > New* y seleccione la opción *Spring Starter Project*.



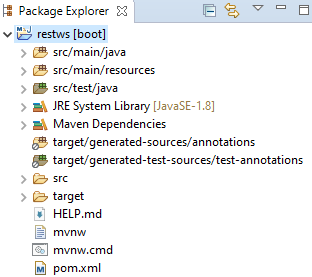
1. Cree el proyecto **restws** con la siguiente configuración y haga clic en Siguiente:



1. Agregue la dependencia Spring Web y haga clic en Siguiente.



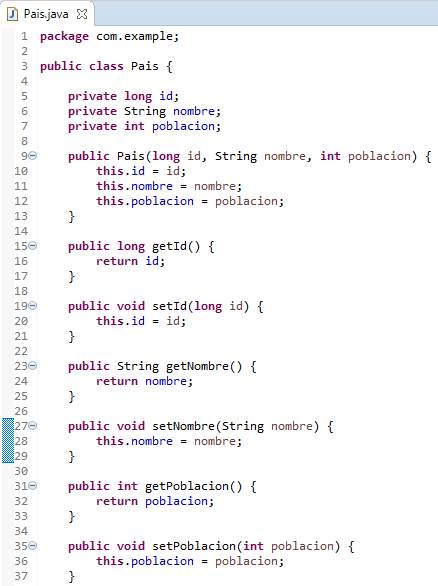
1. En la opción de Site Info, haga clic en Finalizar y verifique su proyecto.



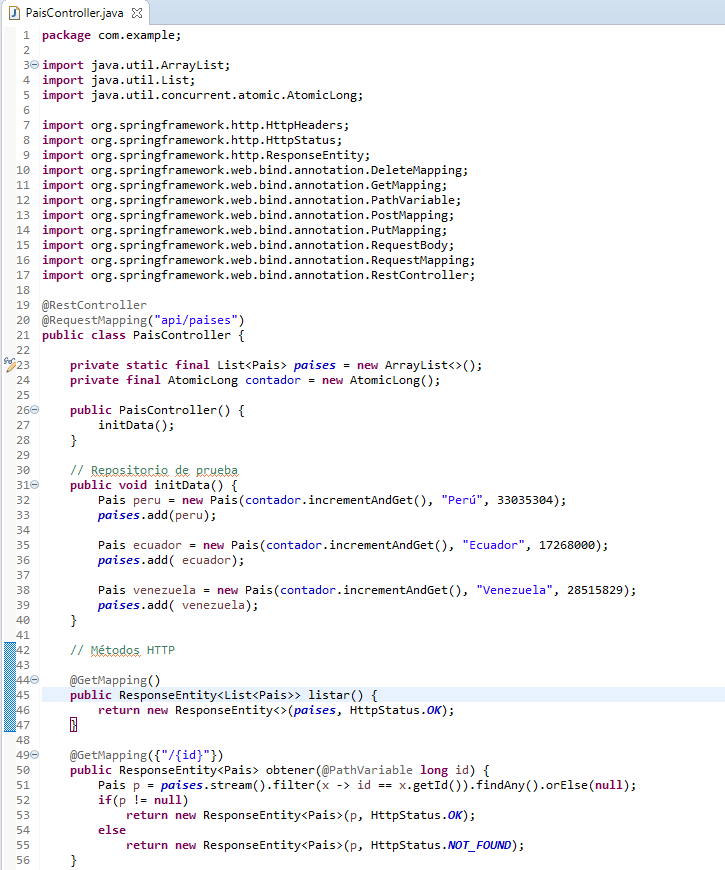
Ahora que ha configurado el proyecto y el sistema de compilación, puede crear su servicio web.

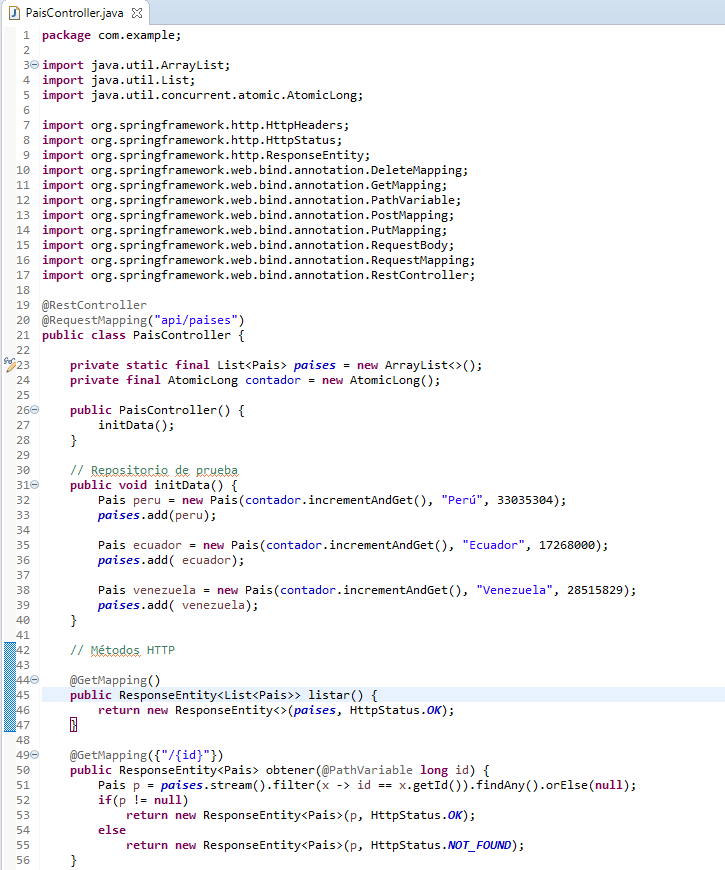
El servicio a desarrollar manejará las solicitudes con todos los métodos HTTP para realizar un CRUD básico. Además, deberá devolver el código de estado 200 para peticiones correctas.

1. Añada la clase java **Pais** para modelar la representación de un país. Esta clase debe contener campos, constructores y descriptores de acceso.



1. En el enfoque de Spring para construir servicios web RESTful, las solicitudes HTTP son manejadas por un controlador. Agregue la clase **PaisController** y añada la anotación @RestController y mapearemos a este recurso con el nombre de **países**. Cada método del CRUD hará uso de los verbos HTTP como GET, POST, PUT y DELETE. En esta clase simularemos una carga inicial de 3 países.

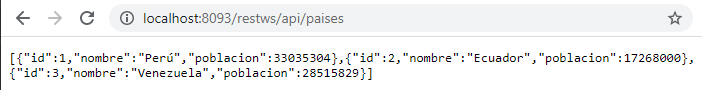




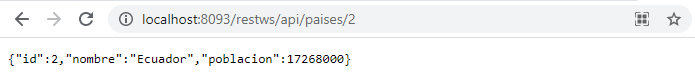


1. Ejecute la aplicación y en su navegador realice las siguientes peticiones:

* <http://localhost:8093/restws/api/paises>

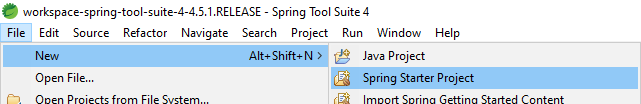


* <http://localhost:8093/restws/api/paises/2>

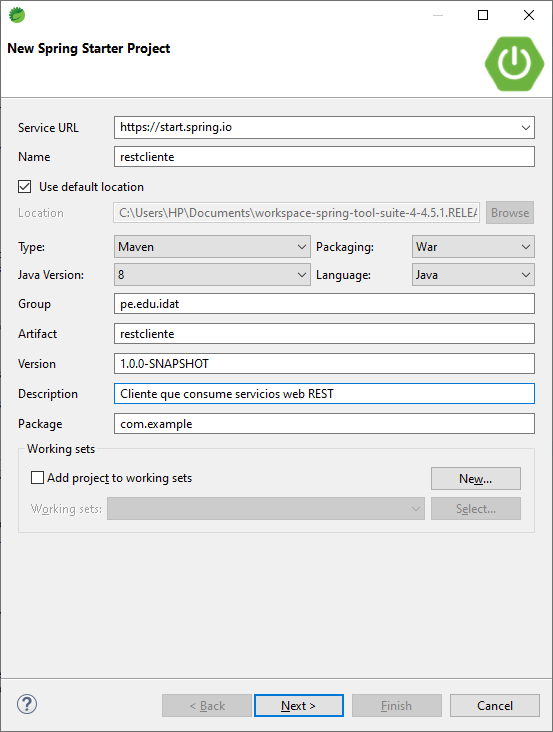


**Consumo de servicio web REST**

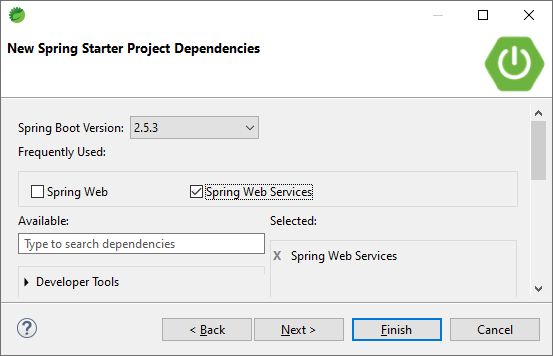
1. En Spring Tool Suite vaya al menú *File > New* y seleccione la opción *Spring Starter Project*.



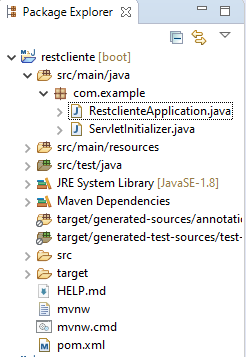
1. Cree el proyecto **restcliente** con la siguiente configuración y haga clic en Siguiente:



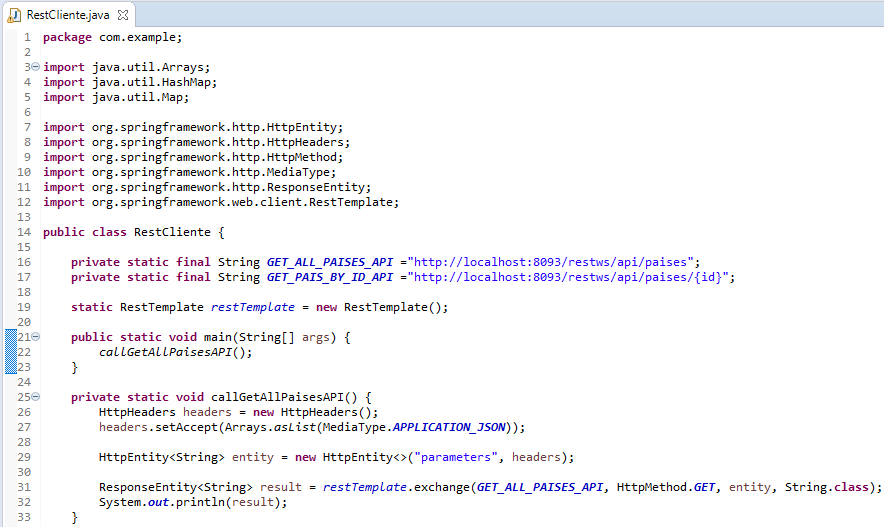
1. Agregue la dependencia Spring Web Services y haga clic en Siguiente.



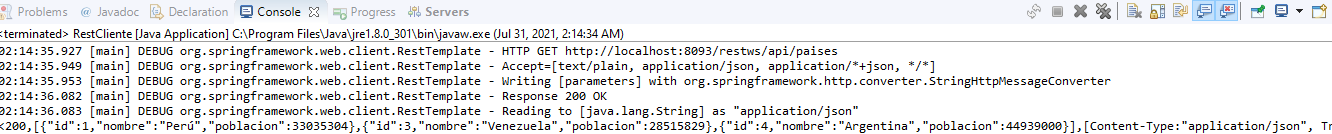
1. En la opción de Site Info, haga clic en Finalizar y verifique su proyecto.



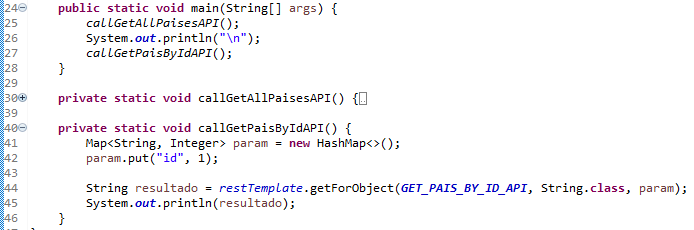
1. Agregue la clase **RestCliente** para realizar el consumo de los servicios web y defina las URL para invocar a los métodos. Agregue el método *callGetAllPaisesAPI* para listar todos los países.



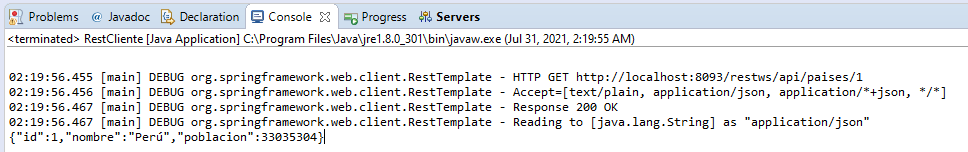
1. Ejecute la clase como JavaApplication y verifique el resultado en la consola.



1. Agregue el método *callGetPaisByIdAPI* y añádalo en el método *main*.

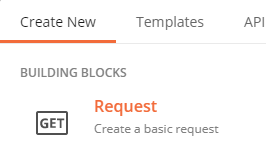


1. Verifique el resultado en la consola.

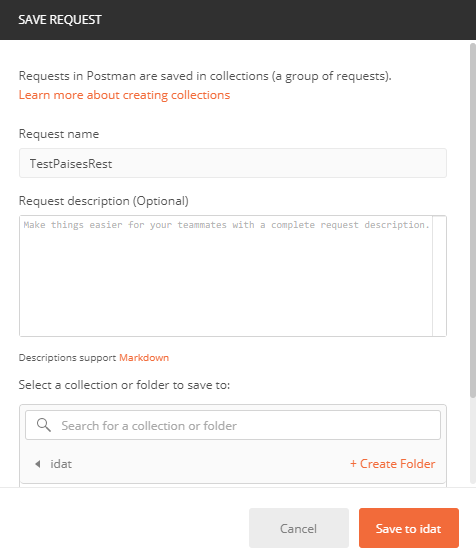


**Pruebas en Postman**

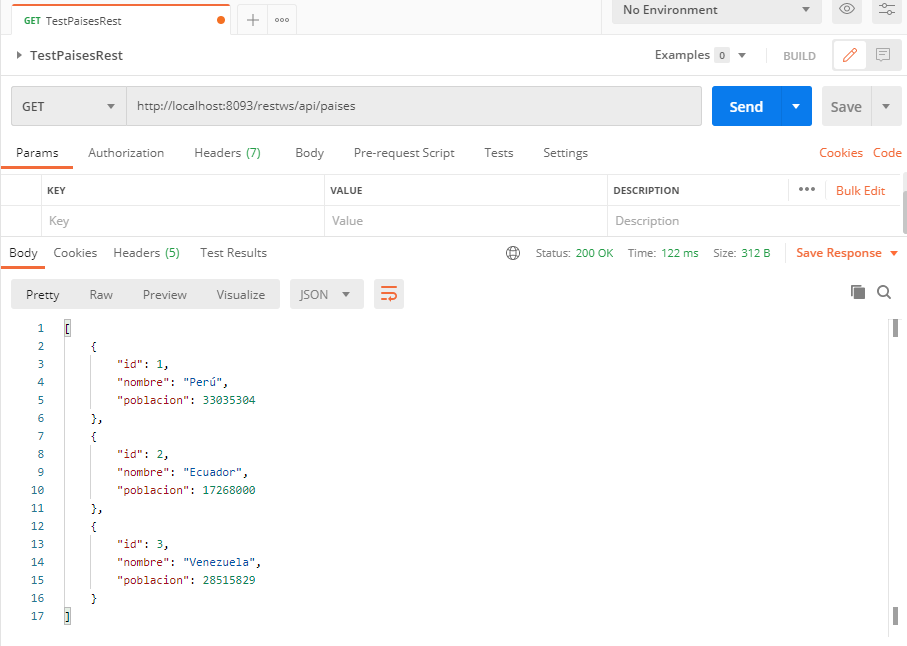
1. En Postman, haga clic en el botón *New* y seleccione la opción *Request*.



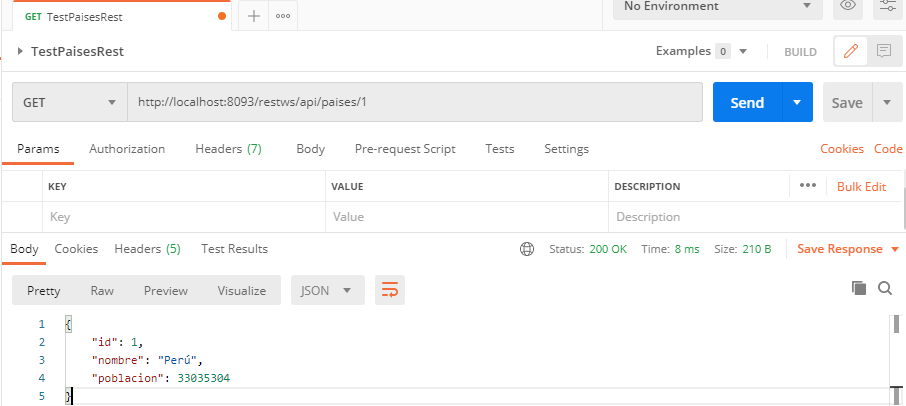
1. En la ventana *Save Request*, ingrese el nombre **TestPaisesRest** y presione OK.



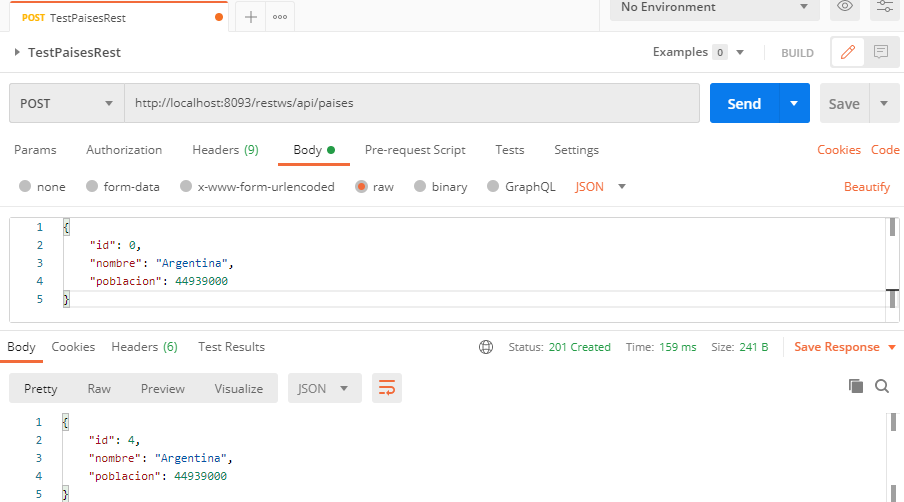
1. Para la petición GET, ingrese la URL <http://localhost:8093/restws/api/paises> y presione en el botón *Send*.



1. Para obtener un solo país, agregue el parámetro id en la misma petición GET y pruebe nuevamente.

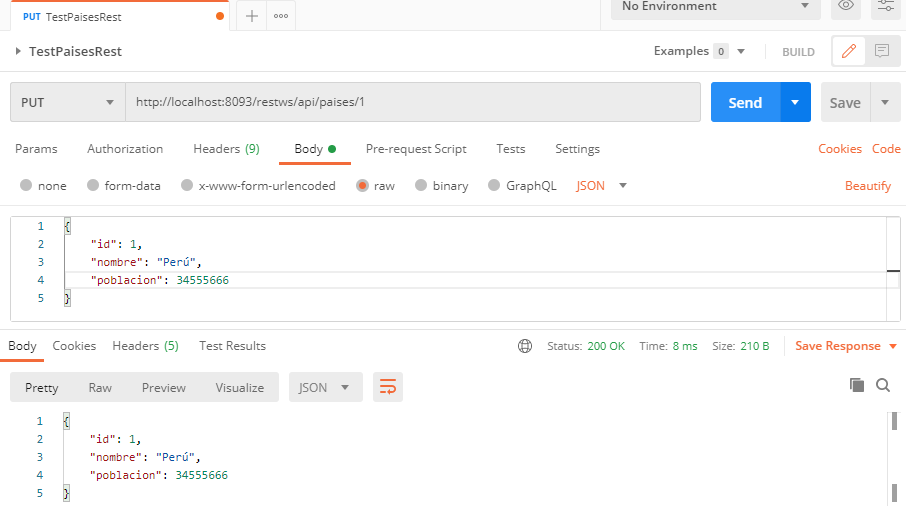


1. Para registrar un nuevo país, seleccione el tipo POST y en la sección *Body* seleccione la opción *Raw* para y el contenido en formato JSON. Luego presione él botón *Send*.

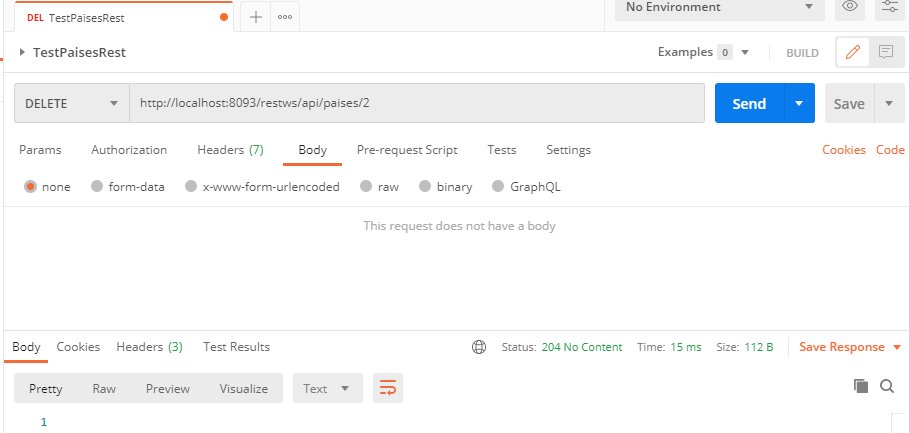


Verifique que se haya generado un nuevo *id*.

1. Para actualizar la información del país Perú, enviaremos el contenido en formato JSON incluyendo el id correspondiente y una población diferente.



1. Para eliminar un país, seleccionamos el método DELETE y agregamos el parámetro *id* sin contenido en el *body*.



Compruebe que el código devuelto es el 204, el cual fue colocado en el método *eliminar* del servicio RES

**Actividad:**

1. Cree una aplicación web que contenga un servicio RESTful que realice un CRUD básico a una entidad (registrar, actualizar, eliminar, listar y obtener por ID). La entidad debe tener como mínimo 5 campos y uno de ellos debe ser un identificador.
2. Cree una aplicación que consuma los métodos creados en la actividad N° 1.
3. En Postman, realice pruebas de 2 servicios REST con la información de Transantiago y analice la información devuelta.

**Documentación**: https://gist.github.com/radutzan/a29aa8fb30b1b866bd0bc44d65a3676e