

EVALUACIÓN Y NUEVAS TENDENCIAS DE

ARQUITECTURA DE SOFTWARE (SI470)

**Practica de Laboratorio**

**Ciclo 2019-2**

**Profesor:**

**Secciones: Todas**

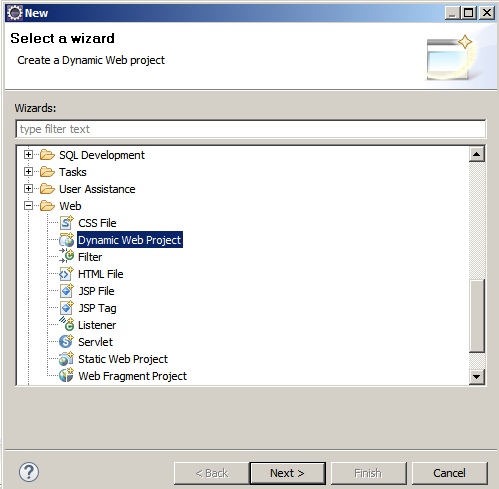
Creación de un API REST

Requisitos:

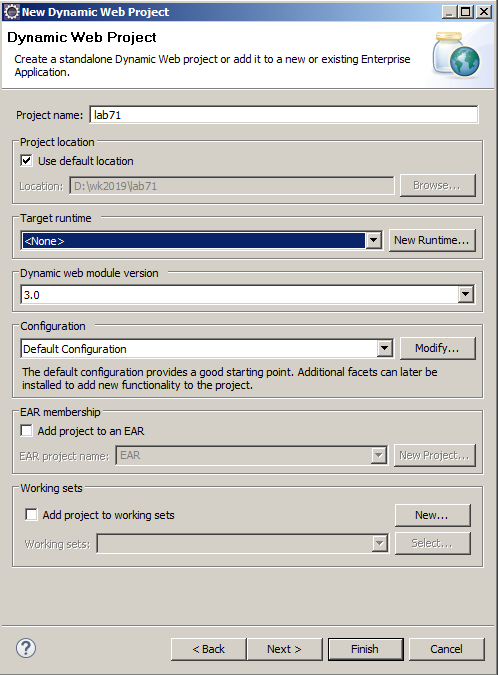
* **Eclipse J2EE**
* **WILDFLY o TOMCAT**

**Crear un Proyecto Dynamic Web Proyect**

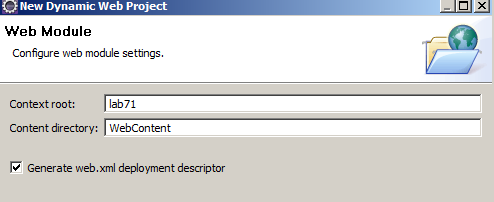
File/New /Other/Web/Dynamic Web Project



Dar nombre al proyecto, en la opción Target runtime seleccionar Apache Tomcat 7.0 si no está indicar el directorio de instalación, si desea desplegarlo en wildfly puede dejarse en NONE.



Asegurarse de marcar la opción Generate Web.xml deployment descriptor



**Estructura del código**

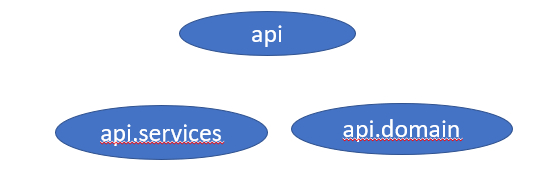
La estructura del código a implementar será de la siguiente manera:

Crear los siguientes paquetes:

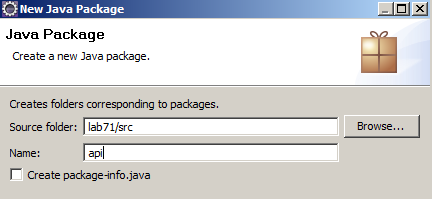
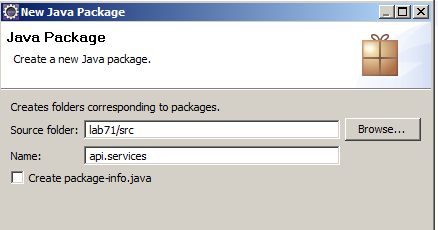
api

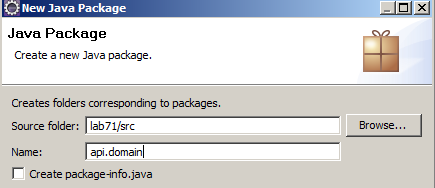
api.services

api.domain

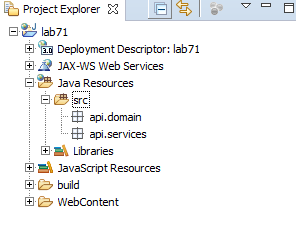


File/New/Package/



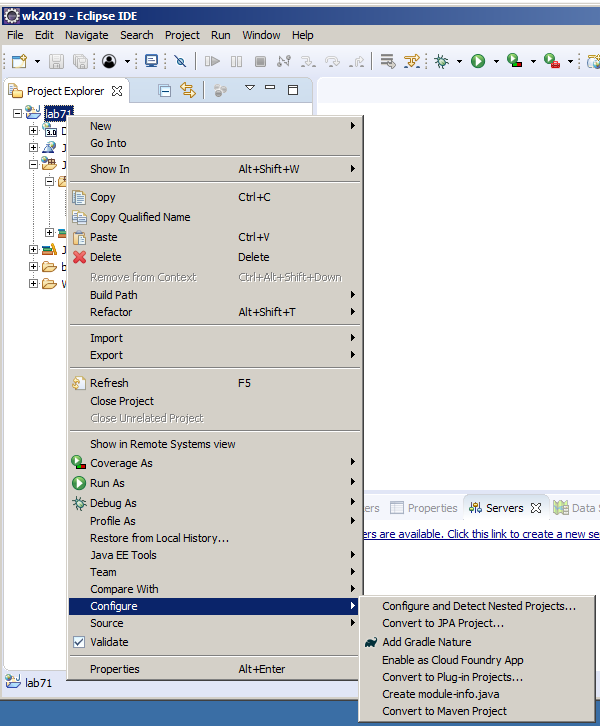
Debe crear la siguiente estructura (en algunas ocasiones no aparece el package api,



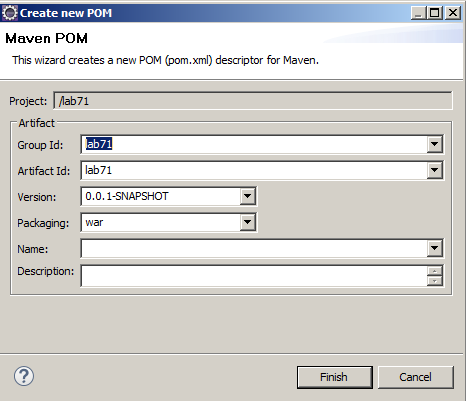
**Convertir proyecto a MAVEN**

Para manejar las dependencias de las referencias necesarias, convertir el proyecto a MAVEN.

Click derecho en el proyecto/configure/convert to maven project



Confirmar, el artefacto a crear con el nombre del proyecto lab71



**Agregar las dependencias del proyecto**

Para que el proyecto incluya las dependencias agregar los siguientes arquetipos en el archivo pom.xml . (Las dependencias están en la carpeta correspondiente en el aula virtual: Unidad3/Labs/API REST). Usar el archivo dependencias:

<dependencies>

<dependency>

<groupId>javax</groupId>

<artifactId>javaee-api</artifactId>

<version>7.0</version>

<scope>provided</scope>

<type>jar</type>

</dependency>

<dependency>

<groupId>org.wildfly.core</groupId>

<artifactId>wildfly-server</artifactId>

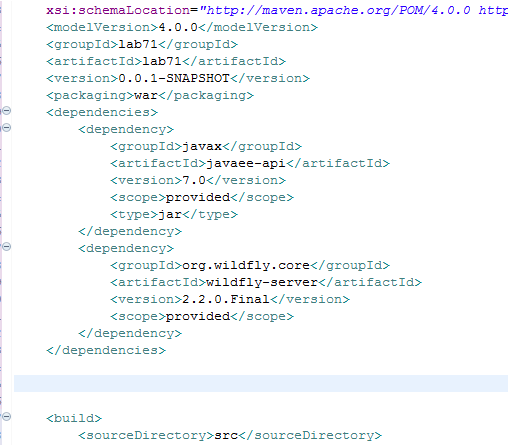
<version>2.2.0.Final</version>

<scope>provided</scope>

</dependency>

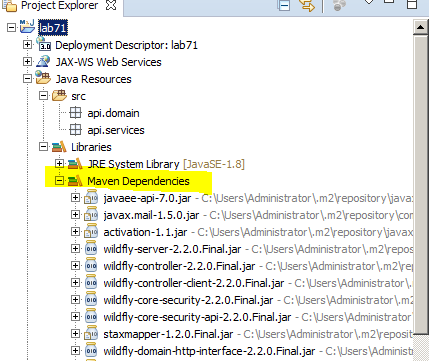
</dependencies>

Agregar al archivo POM.XML antes de <build>



Las dependencias incluyen los arquetipos para jersey y para wildlfy.

Esto permite que se carguen estas dependencias al proyecto.

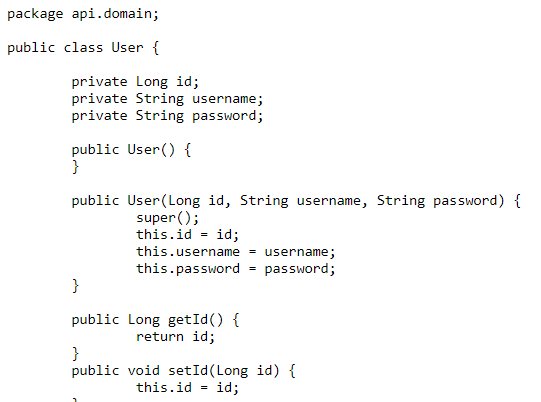


**Implementación del código**

Copiar el código de las siguientes clases en los packages correspondientes

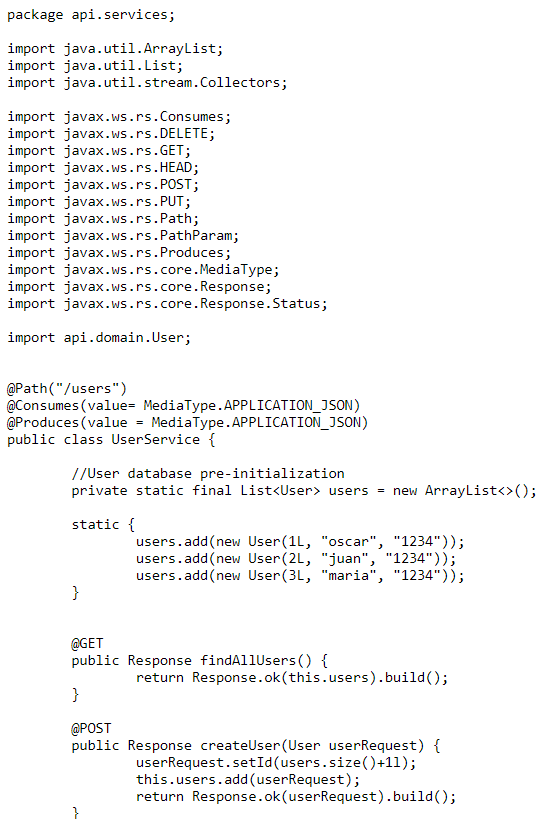
**api.domain**

User.java



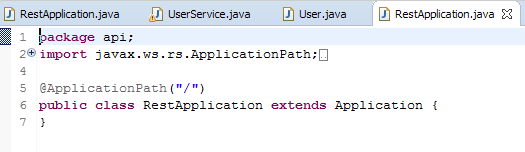
**api.services**

UserService.java

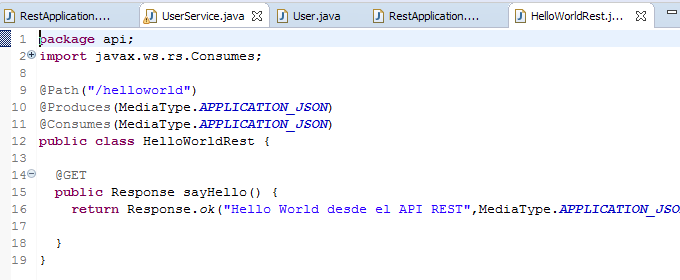


**api**

RestAplication.java (copiarlo en src)



Finalmente copiar la clase HelloWolrdRest.java



**Revisar brevemente cada clase:**

**User** es una clase entidad que debe tener 3 propiedades y métodos set y get, además de los métodos contructor.

**UserService** es la clase que implementa el webservice REST, utiliza los tags Path, Consumes, Produces, GET, PUT, POST,DELETE, HEAD. Esta clase inicialmente precarga en un arreglo 3 registros de 3 personas(id, nombre y clave). Los métodos Update y Delete utilizan expresiones Lambda para hacer búsqueda de un usuario. La clase envia y recibe datos en formato JSON.

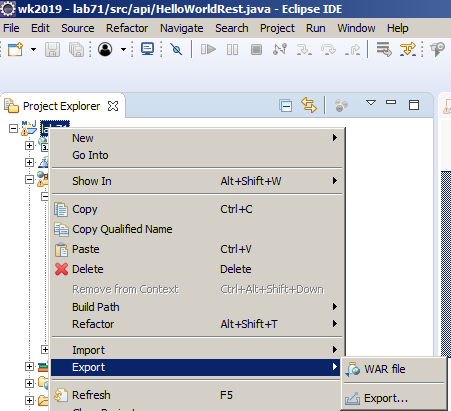
**RestAplication** es una clase que extiende la clase base Aplication que es la que define el punto de entrada de una aplicación, en este caso es la raíz del proyecto /.

**HelloWorldRest** es la típica clase de bienvenida

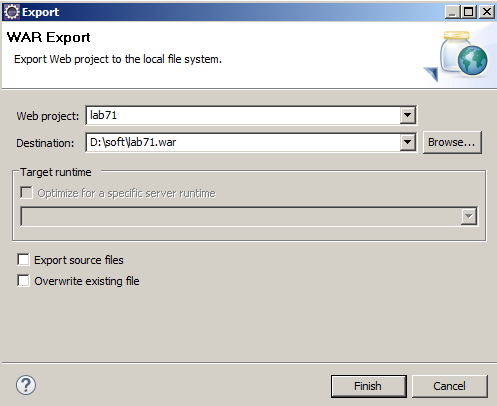
En general se tienen 2 urls principales definidos por la etiqueta Path: “/users” y “/helloworld”

**Exportar el proyecto para desplegarlo en Wildlfy**

Click derecho en el proyecto/Export/War



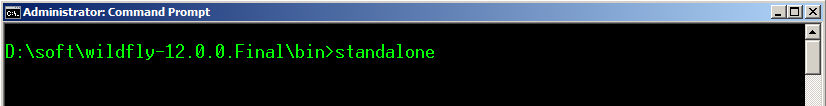
Indicar la carpeta de destino



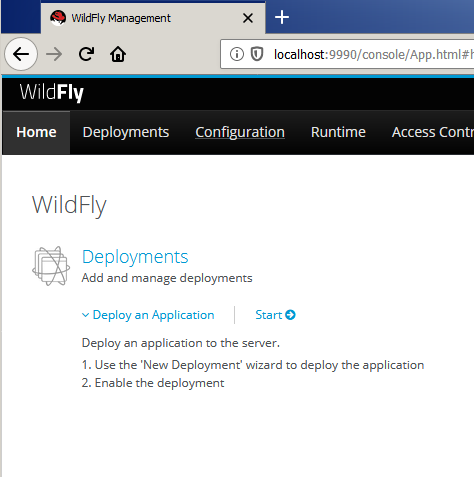
**Iniciar Wildfly y desplegar el war**

Abrir una ventana de comandos e ir a la carpeta wildlfy/bin/

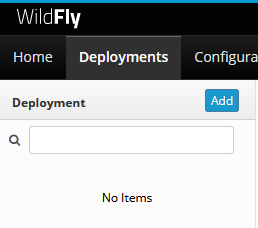
Ejecutar standalone



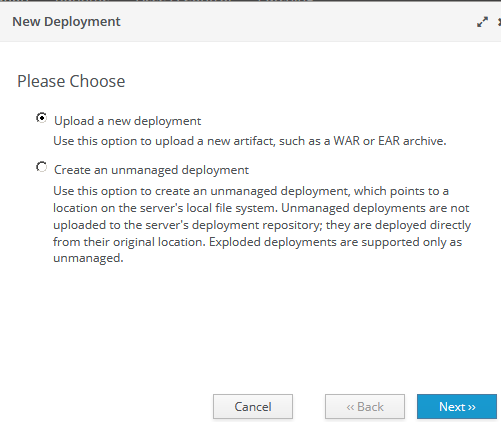
Ir a la consola de administración: <http://localhost:8080>



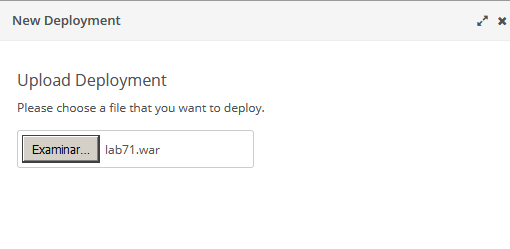
Click en Start/Add



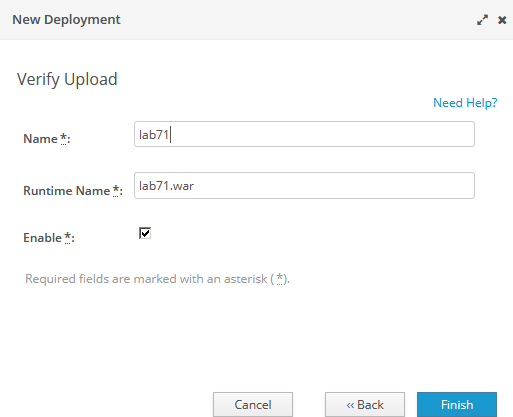
Ir a New Deployment/Upload new deployment



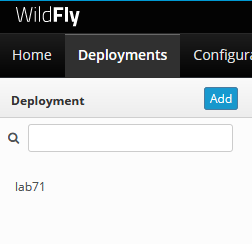
Seleccionar el archivo lab71.war creado en el paso anterior



Indicar nombre de proyecto o aplicación y nombre del paquete a desplegar.



Verificar que aparezca en la lista de aplicaciones desplegadas

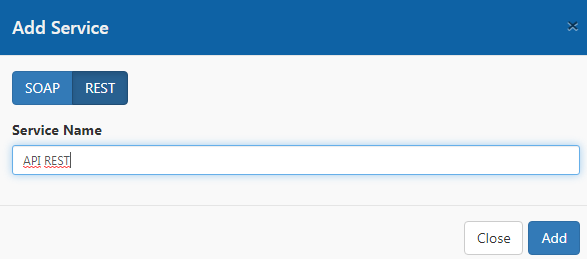


**Prueba del API REST**

Instalar Boomerang en el navegador Chrome

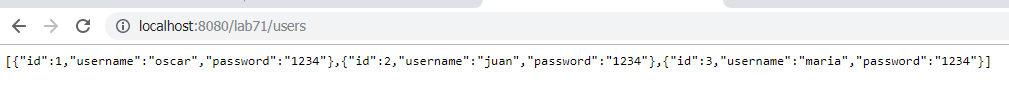


Crear un proyecto en Boomerang, darle nombre API REST



Verificar el funcionamiento del webservice, ingresando a la dirección:

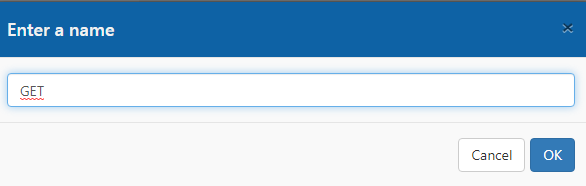
<http://localhost:8080/lab71/users>



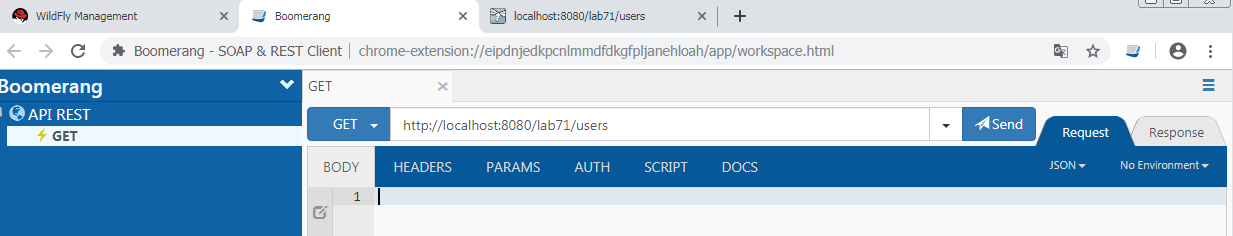
En boomerang crear request llamados GET, PUT, POST,DELETE , HEAD

|  |  |
| --- | --- |
| CRUD en base de datos | Métodos de acceso Webservice REST |
| SELECT | GET |
| INSERT | POST |
| UPDATE | PUT |
| DELETE | DELETE |
|  | HEAD |

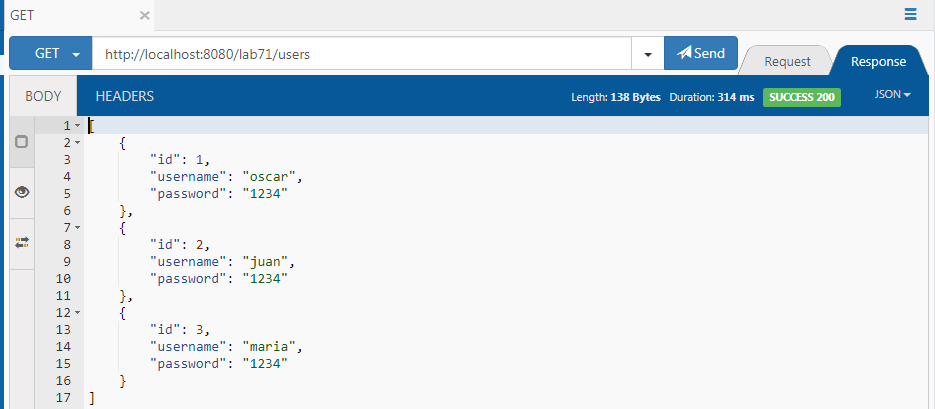
Metodo GET



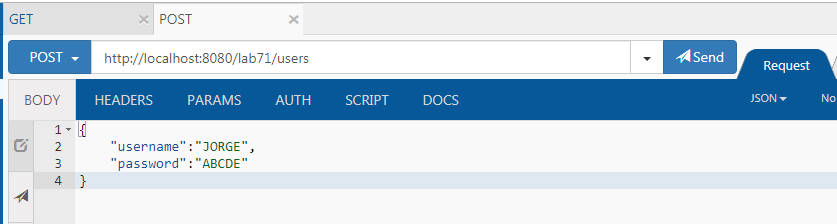
Usar el URL indicado del Webservice <http://localhost:8080/lab71/users>



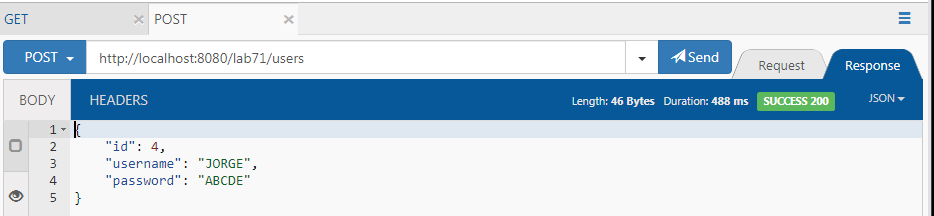
Verificar que la respuesta muestre la siguiente información



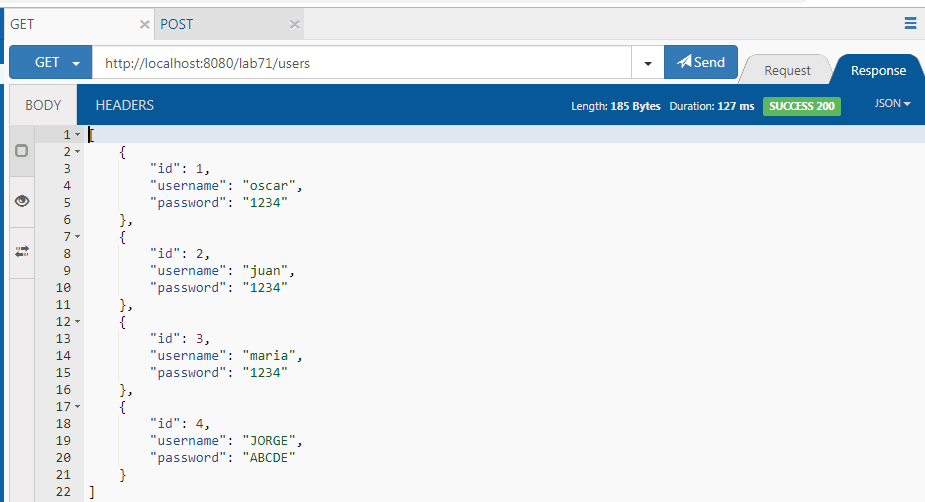
Método POST, ingresar información y crear una estructura JSON con los parámetros username y password en el body



Verificar el response, se genero el id 4

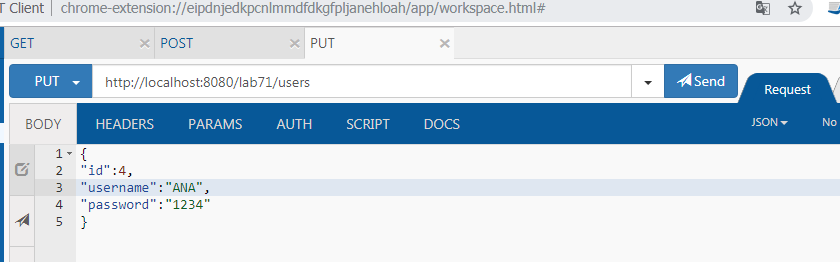


Consultar resultados con método GET, aparece el registro 4

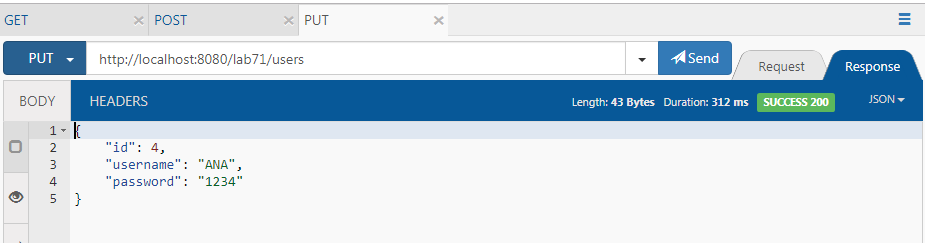


Método PUT (Modificar registro)

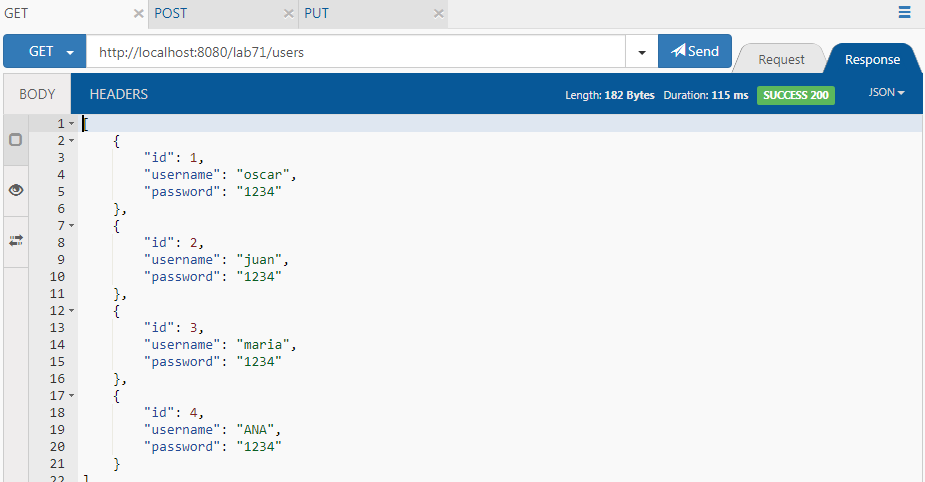
Generar un nuevo request llamado PUT y crear la estructura JSON para modificar el registro 4



Verificar respuesta confirmando cambio del registro 4 a ANA

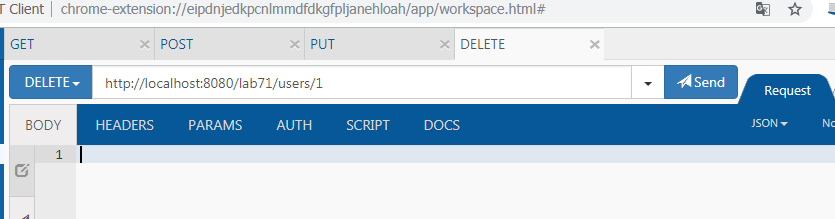


Consultar con GET

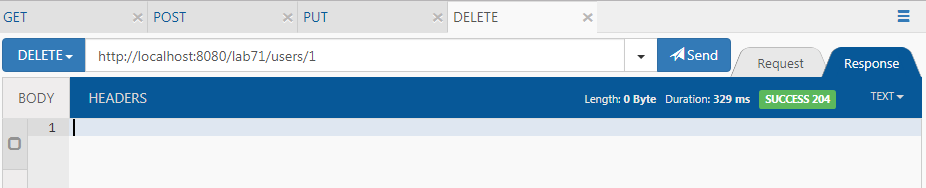


Método DELETE

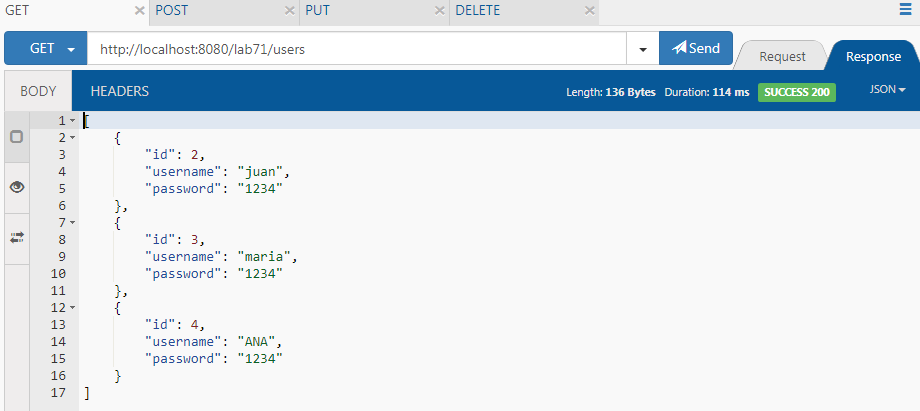
Crear un nuevo request y usar DELETE para eliminar el registro 1, indicar el url con parámetro /1



Verificar la eliminación del registro 1

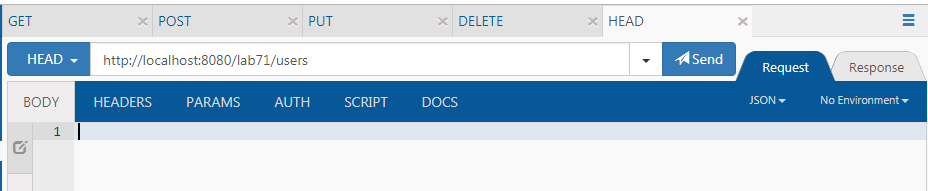


Consultar con GET

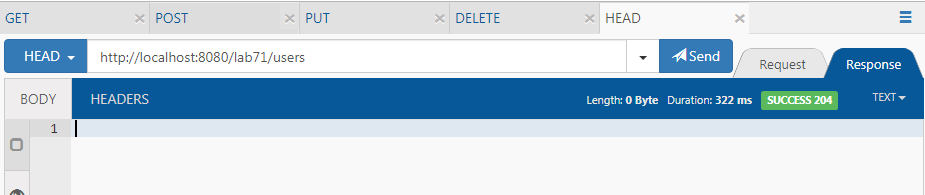


Método HEAD

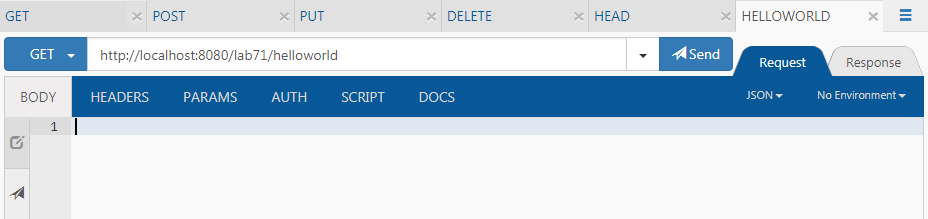
Se utiliza para verificar el funcionamiento del webservice, no se recibe BODY de respuesta, solo status

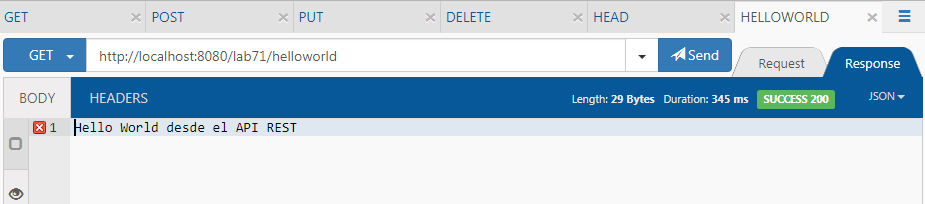


Verificar respuesta y tamaño del body



Finalmente invocar a helloworld





Hacer comparaciones entre la forma de implementar un webservice SOAP

y un webservice REST