

Taller Arquitectura Cliente-Servidor:

Sockets, WebServer y API Rest

Presentado por: Erika Lucia Camacho Dorado

1. Descargar, instalar y correr la clase EchoServerExample que viene en el proyecto strategy-server

Primero se limpio y construyó el proyecto mediante la opción “Clean and Build”.

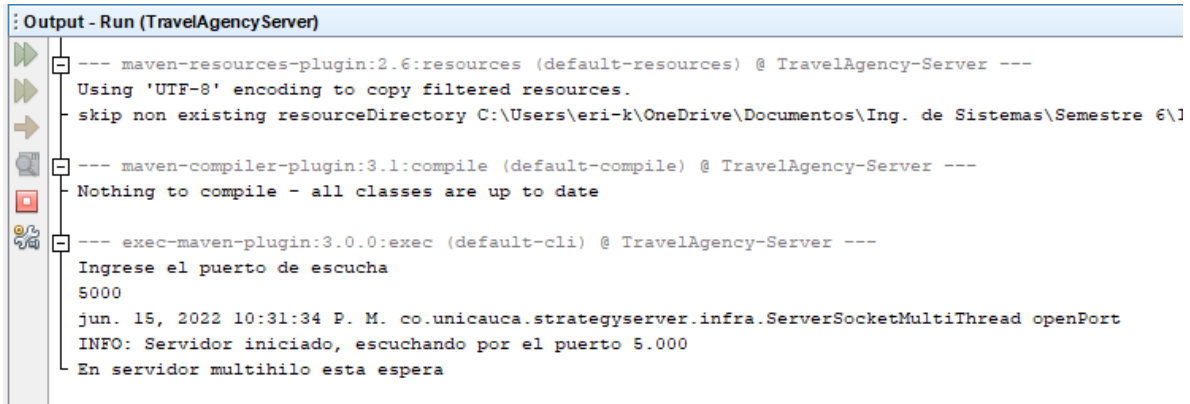
```
Output - Build (strategyserver)
skip non existing resourceDirectory C:\Users\eri-k\OneDrive\Documentos\Ing. de Sistemas\Semestre 6\Ingenieria de software II\Proyectos revision tarea\client-server-evolution-main\strate
--- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ strategyserver ---
No sources to compile
--- maven-surefire-plugin:2.12.4:test (default-test) @ strategyserver ---
No tests to run.
--- maven-jar-plugin:2.4:jar (default-jar) @ strategyserver ---
Building jar: C:\Users\eri-k\OneDrive\Documentos\Ing. de Sistemas\Semestre 6\Ingenieria de software II\Proyectos revision tarea\client-server-evolution-main\strategyserver\target\strate
--- maven-install-plugin:2.4:install (default-install) @ strategyserver ---
Installing C:\Users\eri-k\OneDrive\Documentos\Ing. de Sistemas\Semestre 6\Ingenieria de software II\Proyectos revision tarea\client-server-evolution-main\strategyserver\target\strategyse
Installing C:\Users\eri-k\OneDrive\Documentos\Ing. de Sistemas\Semestre 6\Ingenieria de software II\Proyectos revision tarea\client-server-evolution-main\strategyserver\pom.xml to C:\Us
BUILD SUCCESS
Total time: 16.998 s
Finished at: 2022-06-16T22:17:54-05:00
```

Al ejecutar la clase EchoServerExample se muestra:

```
Output - Run (strategyserver)
cd C:\Users\eri-k\OneDrive\Documentos\Ing. de Sistemas\Semestre 6\Ingenieria de software II\Proye
Scanning for projects...
-----< co.unicauca:strategyserver >-----
Building strategyserver 1.0
-----[ jar ]-----
--- maven-resources-plugin:2.6:resources (default-resources) @ strategyserver ---
Using 'UTF-8' encoding to copy filtered resources.
skip non existing resourceDirectory C:\Users\eri-k\OneDrive\Documentos\Ing. de Sistemas\Semestre
--- maven-compiler-plugin:3.1:compile (default-compile) @ strategyserver ---
Nothing to compile - all classes are up to date
--- exec-maven-plugin:3.0.0:exec (default-cli) @ strategyserver ---
jun. 15, 2022 10:22:13 P. M. co.unicauca.strategyserver.infra.ServerSocketMultiThread openPort
INFO: Servidor iniciado, escuchando por el puerto 5.000
En servidor multihilo esta espera
```

2. Descargar, instalar y correr el AgencyTravelServer como servidor tcp/ip y el AgencyTravel-Client

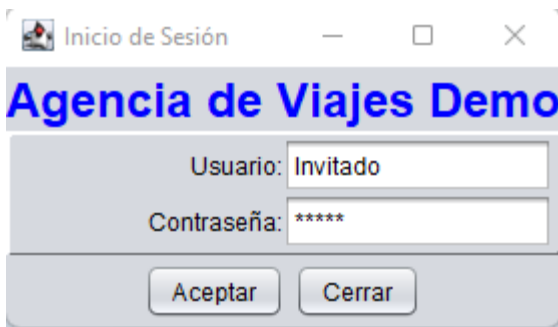
Al correr el AgencyTravelServer:



```
Output - Run (TravelAgencyServer)
--- maven-resources-plugin:2.6:resources (default-resources) @ TravelAgency-Server ---
Using 'UTF-8' encoding to copy filtered resources.
skip non existing resourceDirectory C:\Users\eri-k\OneDrive\Documentos\Ing. de Sistemas\Semestre 6\l
--- maven-compiler-plugin:3.1:compile (default-compile) @ TravelAgency-Server ---
Nothing to compile - all classes are up to date
--- exec-maven-plugin:3.0.0:exec (default-cli) @ TravelAgency-Server ---
Ingrese el puerto de escucha
5000
jun. 15, 2022 10:31:34 P. M. co.unicauca.strategyserver.infra.ServerSocketMultiThread openPort
INFO: Servidor iniciado, escuchando por el puerto 5.000
En servidor multihilo esta espera
```

Al correr el AgencyTravel-Client:

Se muestra la siguiente ventana donde se le da una contraseña y posteriormente se oprime el botón de aceptar.



En la pestaña opciones se selecciona la opción que permite consultar clientes, como se muestra a continuación:

Agencia de viajes

Opciones Informes Configuración Ayuda (Invitado)

Consulta de Clientes

Este ejercicio aplica el patrón cliente/servidor.
La aplicación cliente se conecta al servidor mediante Sockets.
El servidor devuelve el objeto Cliente consultado en formato JSON.
En el backend las cédulas desde 98000001 hasta 98000010.

Número de identificación: 98000001

*Nombres:

*Apellidos:

Dirección:

Celular:

Email:

*Sexo (M, F):

Si la cédula ya se encuentra registrada al oprimir “Buscar” se muestran los datos asociados a ésta.

Agencia de viajes

Opciones Informes Configuración Ayuda (Invitado)

Consulta de Clientes

Este ejercicio aplica el patrón cliente/servidor.
La aplicación cliente se conecta al servidor mediante Sockets.
El servidor devuelve el objeto Cliente consultado en formato JSON.
En el backend las cédulas desde 98000001 hasta 98000010.

Número de identificación: 98000001

*Nombres: Andrea

*Apellidos: Sanchez

Dirección: Calle 14 No 11-12 Popayan

Celular: 3145878752

Email: andrea@hotmail.com

*Sexo (M, F): Femenino

En caso de que la cedula no se encuentre, se permite llenar todos los datos y guardar a un nuevo cliente oprimiendo el botón “Agregar”.

Agencia de viajes

Opciones Informes Configuración Ayuda (Invitado)

Consulta de Clientes

Este ejercicio aplica el patrón cliente/servidor.
La aplicación cliente se conecta al servidor mediante Sockets.
El servidor devuelve el objeto Cliente consultado en formato JSON.
En el backend las cedulas desde 98000001 hasta 98000010.

Número de identificación: 98000012

*Nombres:

*Apellidos:

Dirección:

Celular:

Email:

*Sexo (M, F):

Atención

Cliente no encontrado. Cédula no existe

Agencia de viajes

Opciones Informes Configuración Ayuda (Invitado)

Consulta de Clientes

Este ejercicio aplica el patrón cliente/servidor.
La aplicación cliente se conecta al servidor mediante Sockets.
El servidor devuelve el objeto Cliente consultado en formato JSON.
En el backend las cedulas desde 98000001 hasta 98000010.

Número de identificación: 98000012

*Nombres: Juan

*Apellidos: Certuche

Dirección: calle 5 # 3-30

Celular: 8364548

Email: juan313@gmail.com

*Sexo (M, F): M

Atención

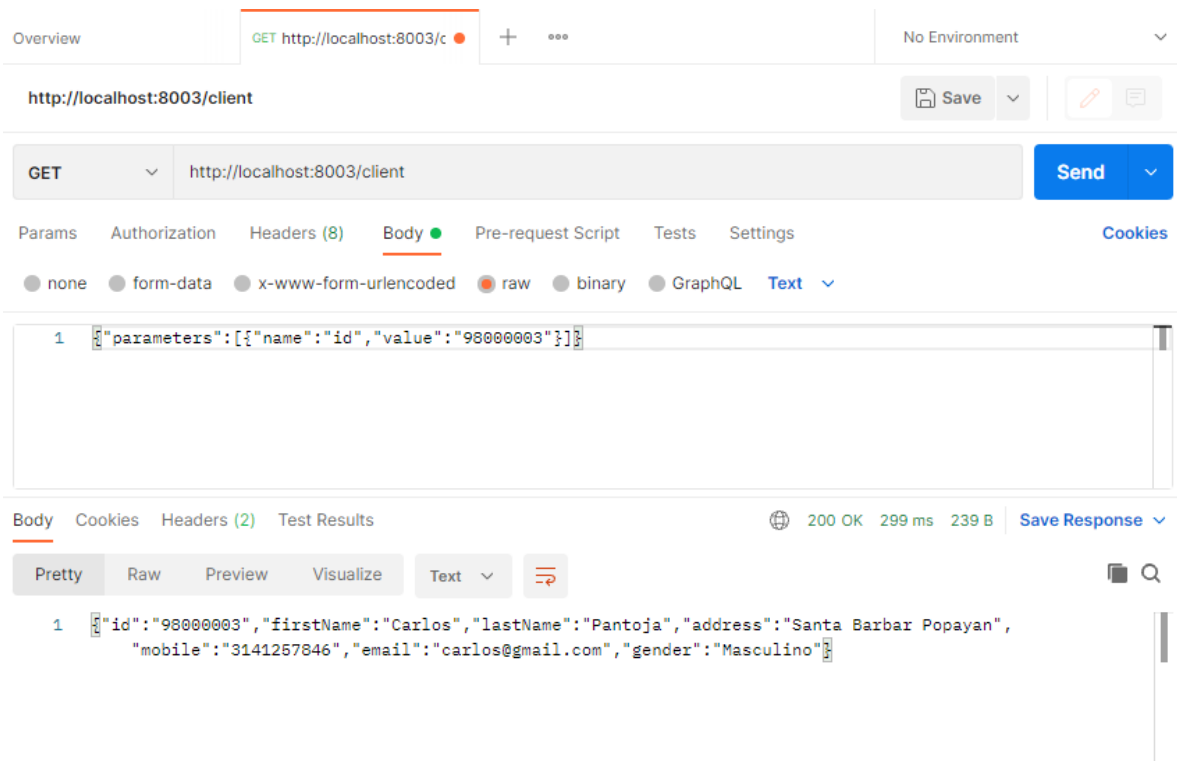
Cliente 98000012 agregado con éxito.

3. Correr el Web Server(infra.web) que vienen en el mismo proyecto AgencyTravelServer. Usar un cliente postman para hacer la consulta, recuerden que el protocolo cambió y ahora sólo van los parámetros.

Al correr el WebServer:

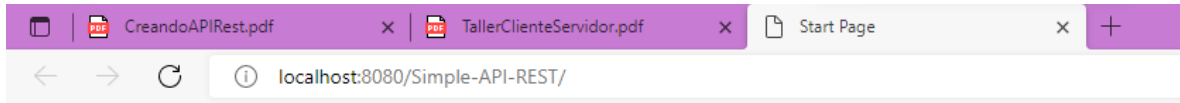
```
Output - Run (WebServer)
-----< co.unicauca.agencia.servidor:TravelAgency-Server >-----
Building TravelAgency-Server 1.0-SNAPSHOT
-----[ jar ]-----
--- maven-resources-plugin:2.6:resources (default-resources) @ TravelAgency-Server ---
Using 'UTF-8' encoding to copy filtered resources.
skip non existing resourceDirectory C:\Users\eri-k\OneDrive\Documentos\Ing. de Sistemas\Semestre 6\
--- maven-compiler-plugin:3.1:compile (default-compile) @ TravelAgency-Server ---
Nothing to compile - all classes are up to date
--- exec-maven-plugin:3.0.0:exec (default-cli) @ TravelAgency-Server ---
Servidor inicializado en el puerto 8003
```

Al hacer la consulta en Postman:



4. Realizar el taller de la API Rest y probar las consultas a través de postman o un Jersey Client. Siguen el paso a paso del taller.

Al correr el proyecto Simple-API-REST:

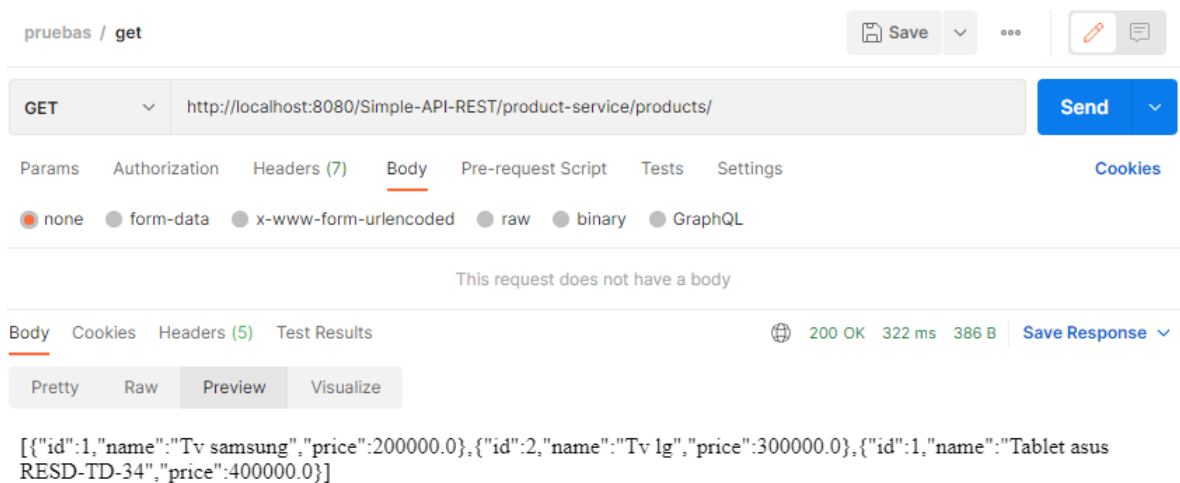


Hello World!

Algunos servicios se probaron usando Postman y también un Jersey Client

Al probar los servicios usando Postman:

- Listar todos los productos.



- Listar un producto en particular, en este ejemplo con id=1

pruebas / get 1 Save ... Edit Comments

GET ▼ http://localhost:8080/Simple-API-REST/product-service/products/1 Send ▼

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies Headers (5) Test Results 200 OK 13 ms 284 B Save Response ▼

Pretty Raw Preview Visualize **JSON** ▼ Copy Search

```
1 {  
2   "id": 1,  
3   "name": "Tv samsung",  
4   "price": 200000.0  
5 }
```

- Eliminar un producto con id=2

pruebas / delete Save ... Edit Comments

DELETE ▼ http://localhost:8080/Simple-API-REST/product-service/products/2 Send ▼

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

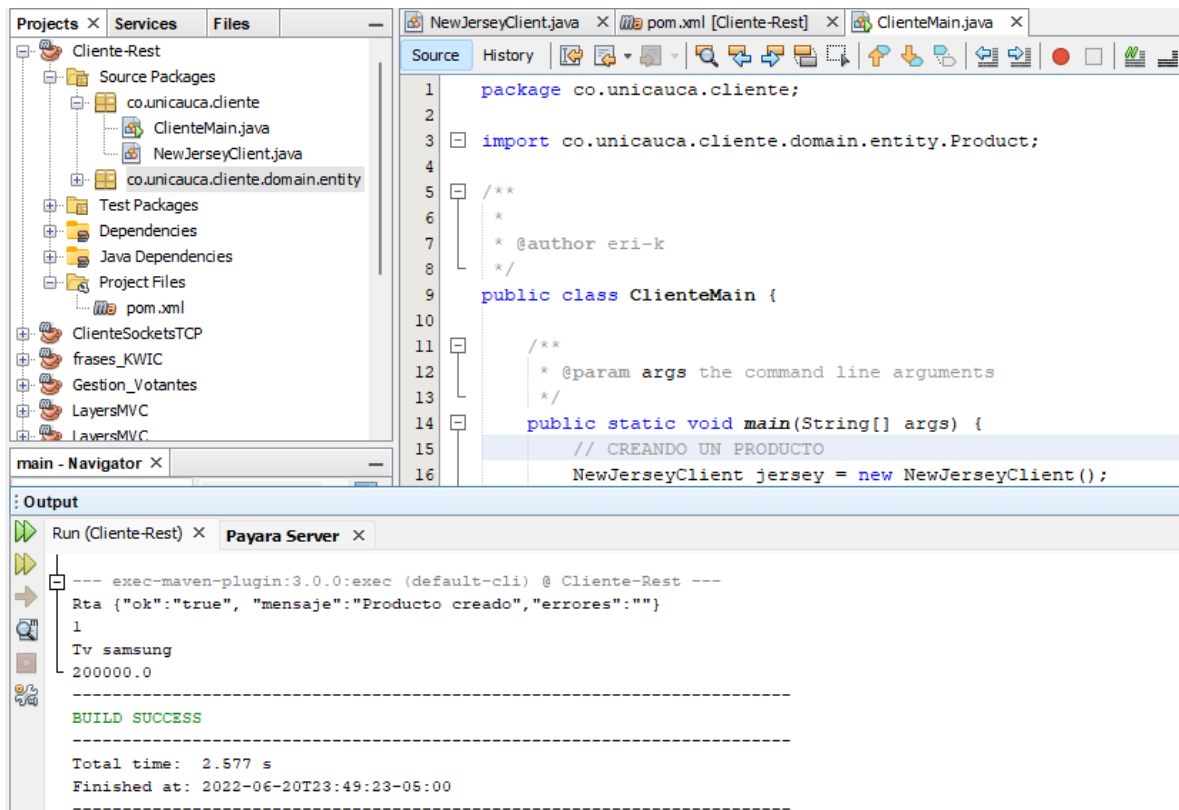
Body Cookies Headers (5) Test Results 200 OK 20 ms 289 B Save Response ▼

Pretty Raw Preview Visualize **Text** ▼ Copy Search

```
1 {"ok": "true", "mensaje": "Producto borrado", "errores": ""}
```

Creando la aplicación cliente (Usando Jersey Client)

Al ejecutar el ClienteMain con los servicios crear y buscar un producto.



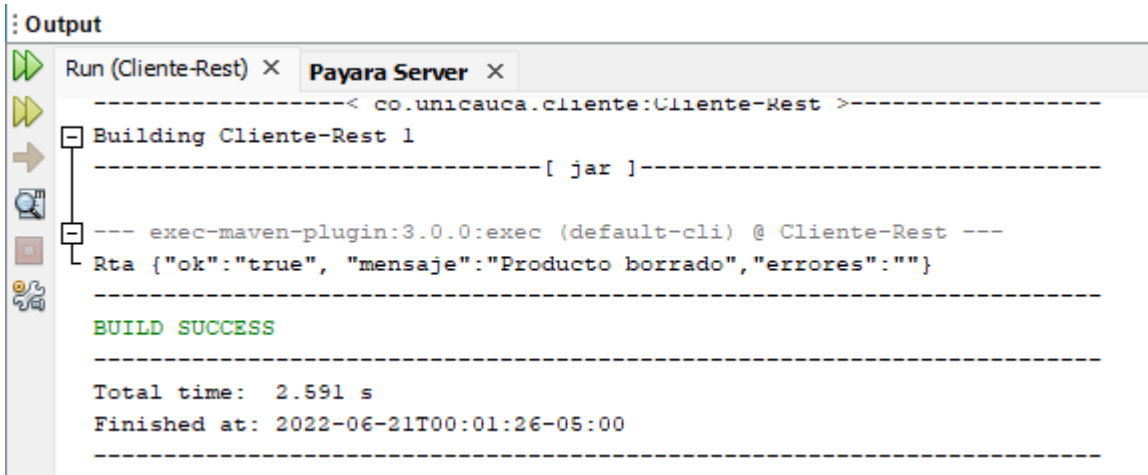
```
1 package co.unicauca.cliente;
2
3 import co.unicauca.cliente.domain.entity.Product;
4
5 /**
6  *
7  * @author eri-k
8  */
9 public class ClienteMain {
10
11     /**
12     * @param args the command line arguments
13     */
14     public static void main(String[] args) {
15         // CREANDO UN PRODUCTO
16         NewJerseyClient jersey = new NewJerseyClient();
```

Output

```
Run (Cliente-Rest) x Payara Server x
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Cliente-Rest ---
Rta {"ok":"true", "mensaje":"Producto creado","errores":""}
1
Tv samsung
200000.0
-----
BUILD SUCCESS
-----
Total time: 2.577 s
Finished at: 2022-06-20T23:49:23-05:00
-----
```

Probando el servicio eliminar producto

```
//ELIMINAR UN PRODUCTO
Product product2 = jersey.findById(Product.class, "6");
String respuesta = jersey.remove(product2.getId().toString());
System.out.println("Rta " + respuesta);
```

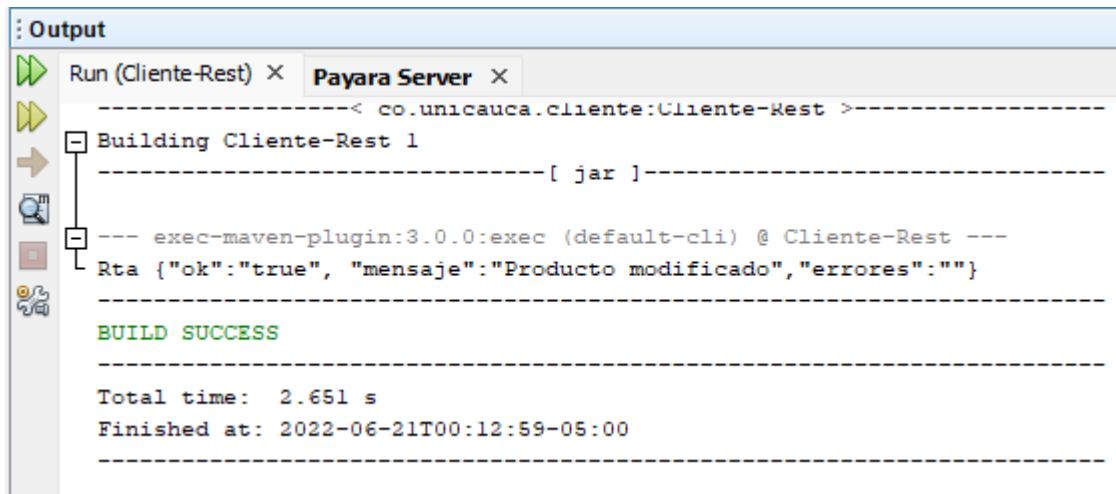


Output

```
Run (Cliente-Rest) x Payara Server x
-----< co.unicauca.cliente:Cliente-Rest >-----
Building Cliente-Rest 1
-----[ jar ]-----
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Cliente-Rest ---
Rta {"ok":"true", "mensaje":"Producto borrado","errores":""}
-----
BUILD SUCCESS
-----
Total time: 2.591 s
Finished at: 2022-06-21T00:01:26-05:00
-----
```


Probando el servicio editar producto

```
//EDITAR UN PRODUCTO
Product product = jersey.findById(Product.class, "2");
product.setName("Tv Caixun");
product.setPrice(25000000d);
String res = jersey.update_JSON(product);
System.out.println("Rta " + res);
```




```
Output
Run (Cliente-Rest) X Payara Server X
-----< co.unicauca.cliente:Cliente-Rest >-----
Building Cliente-Rest 1
-----[ jar ]-----
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Cliente-Rest ---
Rta {"ok":"true", "mensaje":"Producto modificado","errores":""}
-----
BUILD SUCCESS
-----
Total time: 2.651 s
Finished at: 2022-06-21T00:12:59-05:00
-----
```

5. Hacer una API Rest para el AgencyTravelServer. Probarla desde un Jersey Client o a través de Postman. **En este caso se probó desde un Jersey Client.**

Clase ClienteMain (Dentro del paquete co.unicauca.travelagency.cliente en el proyecto TravelAgencyServer, en este paquete también se encuentra la clase NewJerseyClient):

```
public class ClienteMain {  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // CREANDO UN CLIENTE  
        NewJerseyClient jersey = new NewJerseyClient();  
        String rta = jersey.create_JSON(new Customer("98000012", "Sebastian", "Arango",  
            "Calle 8 No 11-14 Popayan", "3154786950", "sebasar@hotmail.com", "Masculino"));  
        System.out.println("Resultado creación cliente");  
        System.out.println("Rta " + rta);  
        System.out.println("");  
        // BUSCANDO UN CLIENTE  
        Customer cust2 = jersey.findById(Customer.class, "98000004");  
        System.out.println("Resultado búsqueda de cliente");  
        System.out.println("Id: " + cust2.getId());  
        System.out.println("Nombre: " + cust2.getFirstName());  
        System.out.println("Apellido: " + cust2.getLastName());  
        System.out.println("Dirección: " + cust2.getAddress());  
        System.out.println("Telefono movil: " + cust2.getMobile());  
        System.out.println("Email: " + cust2.getEmail());  
        System.out.println("Género: " + cust2.getGender());  
        System.out.println("");  
        //ELIMINAR UN CLIENTE  
        Customer cust3 = jersey.findById(Customer.class, "98000009");  
        String respuesta = jersey.remove(cust3.getId());  
        System.out.println("Resultado eliminacion cliente");  
        System.out.println("Rta " + respuesta);  
        System.out.println("");  
        //EDITAR UN CLIENTE  
        Customer cust4 = jersey.findById(Customer.class, "98000004");  
        cust4.setFirstName("Juan");  
        cust4.setLastName("Perez");  
        cust4.setAddress("calle 5 No 8-30");  
        cust4.setGender("Masculino");  
        cust4.setEmail("juanp@gmail.com");  
        cust4.setMobile("3203456786");  
        String res = jersey.update_JSON(cust4);  
        System.out.println("Resultado actualización cliente");  
        System.out.println("Rta " + res);  
    }  
}
```

Resultado al probar los servicios



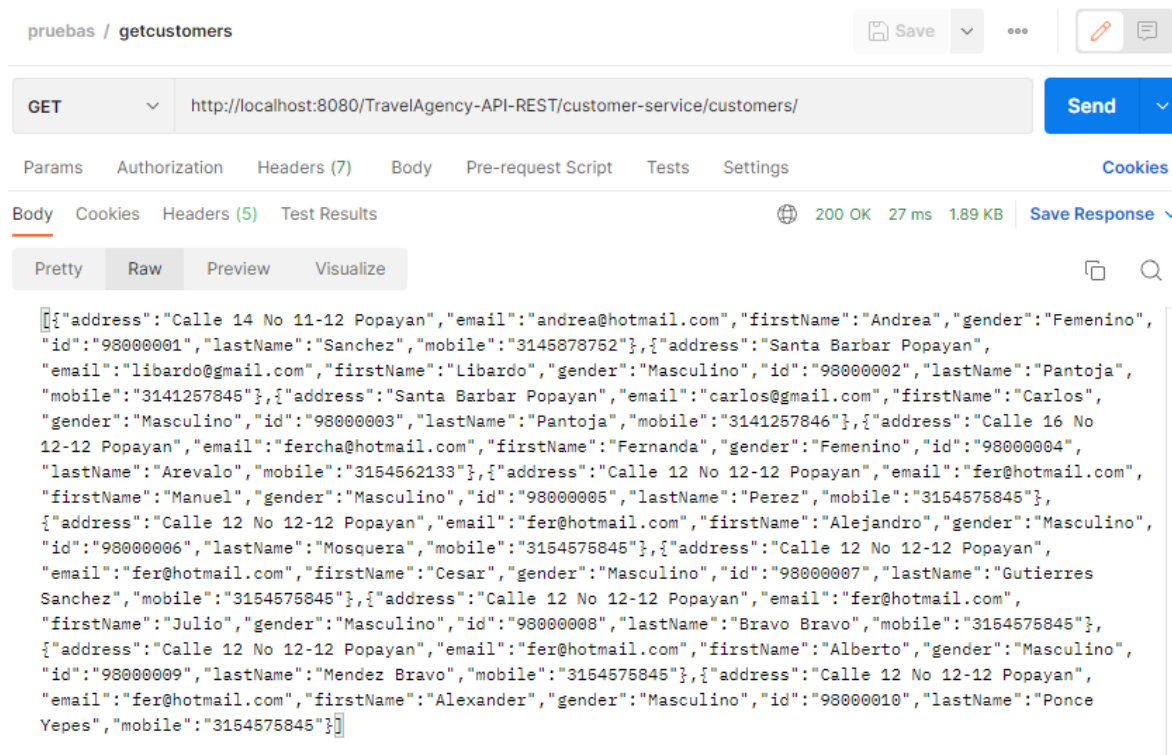
```
Output
Payara Server x Run (ClienteMain) x
-----[ jar ]-----
--- exec-maven-plugin:3.0.0:exec (default-cli) @ TravelAgency-Server ---
Resultado creación cliente
Rta {"ok":"true", "mensaje":"Cliente creado","errores":""}

Resultado búsqueda de cliente
Id: 98000004
Nombre: Juan
Apellido: Perez
Dirección: calle 5 No 8-30
Telefono movil: 3203456786
Email: juanp@gmail.com
Género: calle 5 No 8-30

Resultado eliminacion cliente
Rta {"ok":"true", "mensaje":"Cliente borrado","errores":""}

Resultado actualización cliente
Rta {"ok":"true", "mensaje":"Cliente modificado","errores":""}
-----
BUILD SUCCESS
-----
```

Para probar el findAll se utilizó Postman obteniendo los siguientes resultados:



pruebas / getcustomers

GET http://localhost:8080/TravelAgency-API-REST/customer-service/customers/ Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (5) Test Results 200 OK 27 ms 1.89 KB Save Response

Pretty Raw Preview Visualize

```
[{"address": "Calle 14 No 11-12 Popayan", "email": "andrea@hotmail.com", "firstName": "Andrea", "gender": "Femenino", "id": "98000001", "lastName": "Sanchez", "mobile": "3145878752"}, {"address": "Santa Barbar Popayan", "email": "libardo@gmail.com", "firstName": "Libardo", "gender": "Masculino", "id": "98000002", "lastName": "Pantoja", "mobile": "3141257845"}, {"address": "Santa Barbar Popayan", "email": "carlos@gmail.com", "firstName": "Carlos", "gender": "Masculino", "id": "98000003", "lastName": "Pantoja", "mobile": "3141257846"}, {"address": "Calle 16 No 12-12 Popayan", "email": "fercha@hotmail.com", "firstName": "Fernanda", "gender": "Femenino", "id": "98000004", "lastName": "Arevalo", "mobile": "3154562133"}, {"address": "Calle 12 No 12-12 Popayan", "email": "fer@hotmail.com", "firstName": "Manuel", "gender": "Masculino", "id": "98000005", "lastName": "Perez", "mobile": "3154575845"}, {"address": "Calle 12 No 12-12 Popayan", "email": "fer@hotmail.com", "firstName": "Alejandro", "gender": "Masculino", "id": "98000006", "lastName": "Mosquera", "mobile": "3154575845"}, {"address": "Calle 12 No 12-12 Popayan", "email": "fer@hotmail.com", "firstName": "Cesar", "gender": "Masculino", "id": "98000007", "lastName": "Gutierrez Sanchez", "mobile": "3154575845"}, {"address": "Calle 12 No 12-12 Popayan", "email": "fer@hotmail.com", "firstName": "Julio", "gender": "Masculino", "id": "98000008", "lastName": "Bravo Bravo", "mobile": "3154575845"}, {"address": "Calle 12 No 12-12 Popayan", "email": "fer@hotmail.com", "firstName": "Alberto", "gender": "Masculino", "id": "98000009", "lastName": "Mendez Bravo", "mobile": "3154575845"}, {"address": "Calle 12 No 12-12 Popayan", "email": "fer@hotmail.com", "firstName": "Alexander", "gender": "Masculino", "id": "98000010", "lastName": "Ponce Yepes", "mobile": "3154575845"}]
```

Enlace del código punto 5.

<https://github.com/luciac1103/Taller-arquitectura-Cliente-Servidor>