0ª Emisión

DIPLOMADO

# Desarrollo de Sistemas con Tecnología Java

## Módulo 10

## Servicios Web y Beans Empresariales

*Ing. Jorge Alberto Montalvo Olvera*

DGTIC

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Dirección General de Cómputo y de Tecnologías de información y Comunicación
Dirección de Docencia en TIC

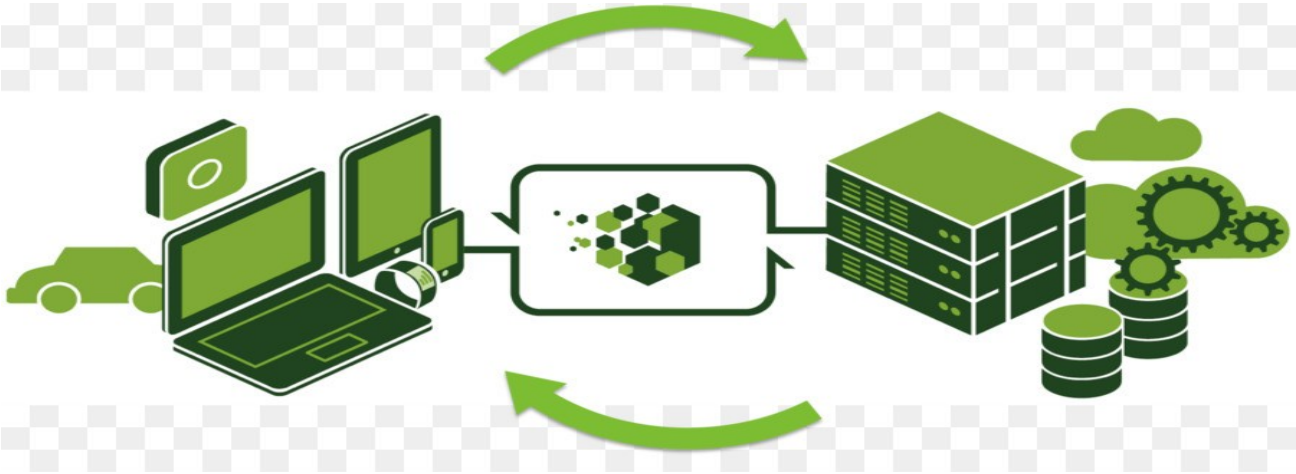
Educación Continua
1971 - 2021

# TEMARIO

- Introducción
  - Servicios Web
  - Rest
- Servicios Web RESTful
  - Construcción
  - Consumo Servicios Web RESTful
- Beans Empresariales
- El contenedor embebido de Beans Empresariales
- Invocación asíncrona de métodos en los Beans de Sesión.

# Servicios Web

- Un Web Service, o Servicio Web, es un método de comunicación entre dos plataformas diferentes en una red.
- Es una colección de protocolos abiertos y estándares usados para intercambiar datos entre aplicaciones o sistemas.

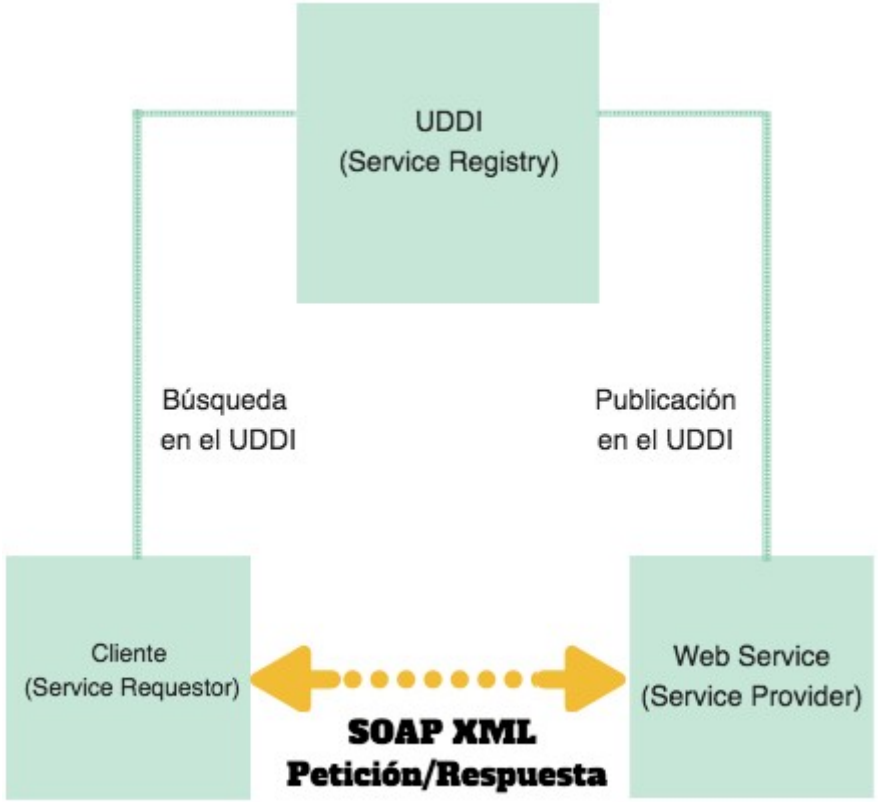


# Servicios Web

- La base de comunicación entre servicios web es el lenguaje XML y el protocolo HTTP.

# Servicio Web

- El protocolo más simple para el intercambio de información entre equipos de cómputo es XML-RPC, que emplea XML para llevar a cabo RPCs. RPC, Remote Procedure Call, es un protocolo de red que permite a un programa a ejecutar código en una máquina remota.



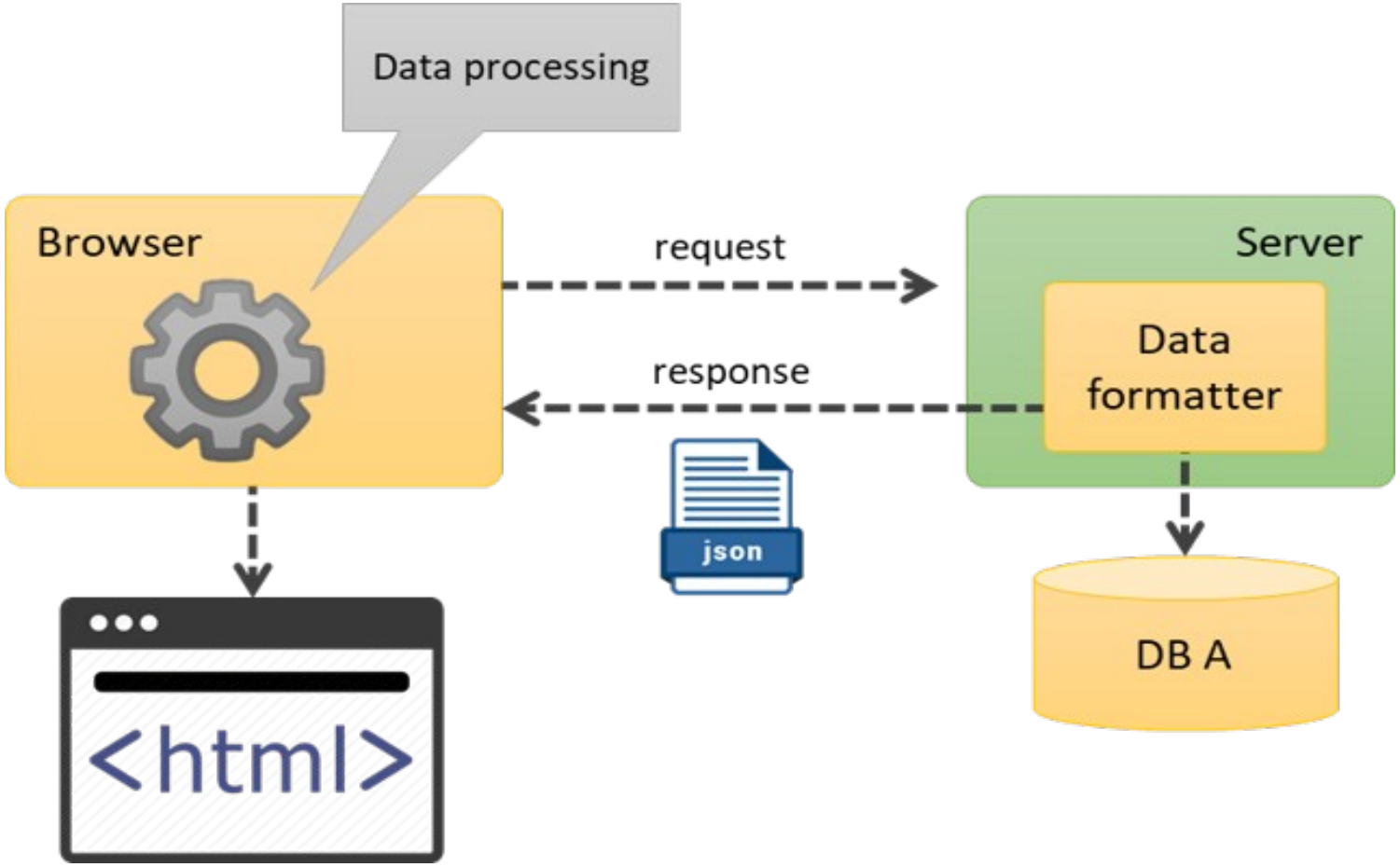
# Servicio Web

- Los servicios web estandarizados funcionan con los siguientes componentes:
  - SOAP - Simple Object Access Protocol
  - WSDL - Web Services Description Language
  - UDDI - Universal Description, Discovery and Integration

# Rest

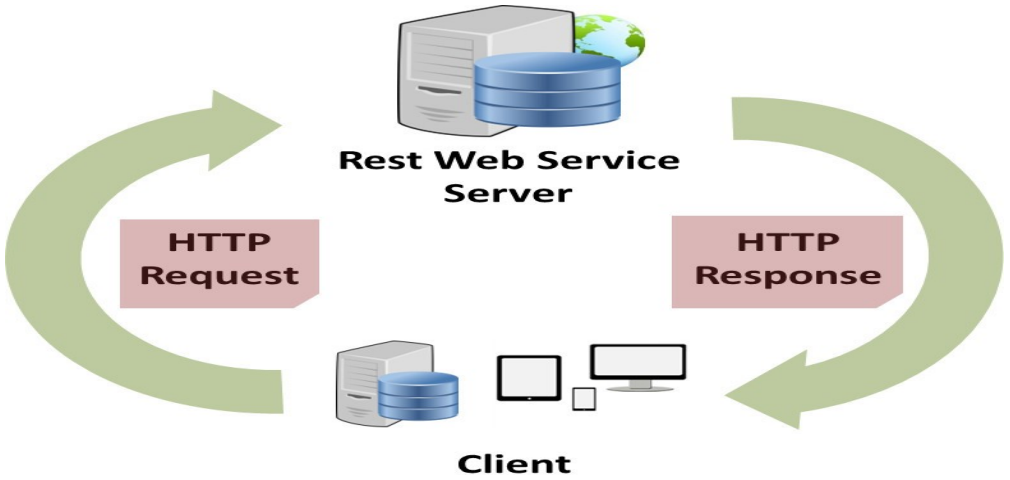
- Rest es una arquitectura para aplicaciones basadas en redes, sus siglas significan REpresentational State Transfer.
  - Se diseñó pensando en ser simple, con ello se lograría una rápida adopción del usuario y un desarrollo rápido.
  - Es un estilo arquitectónico diseñado para y sobre un sistema distribuido particular, la Web.

# Rest


Data processing
Browser
request
Server
Data
formatter
response
json
DB A
<html>

# Servicio Web RESTful

- Servicio Web RESTful es aquél servicio web que está basado en la arquitectura REST.
- Los 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.



# Servicio Web RESTful

- Usualmente los RESTful web service tienen estas características:
  - Están asociados a información
  - Permiten listar, crear, leer, actualizar y borrar información
  - Para las operaciones anteriores necesitan una URL y un método HTTP para accederlas
  - Usualmente regresan la información en formato JSON.
  - Retornan códigos de respuesta HTML, por ejemplo 200, 201, 404

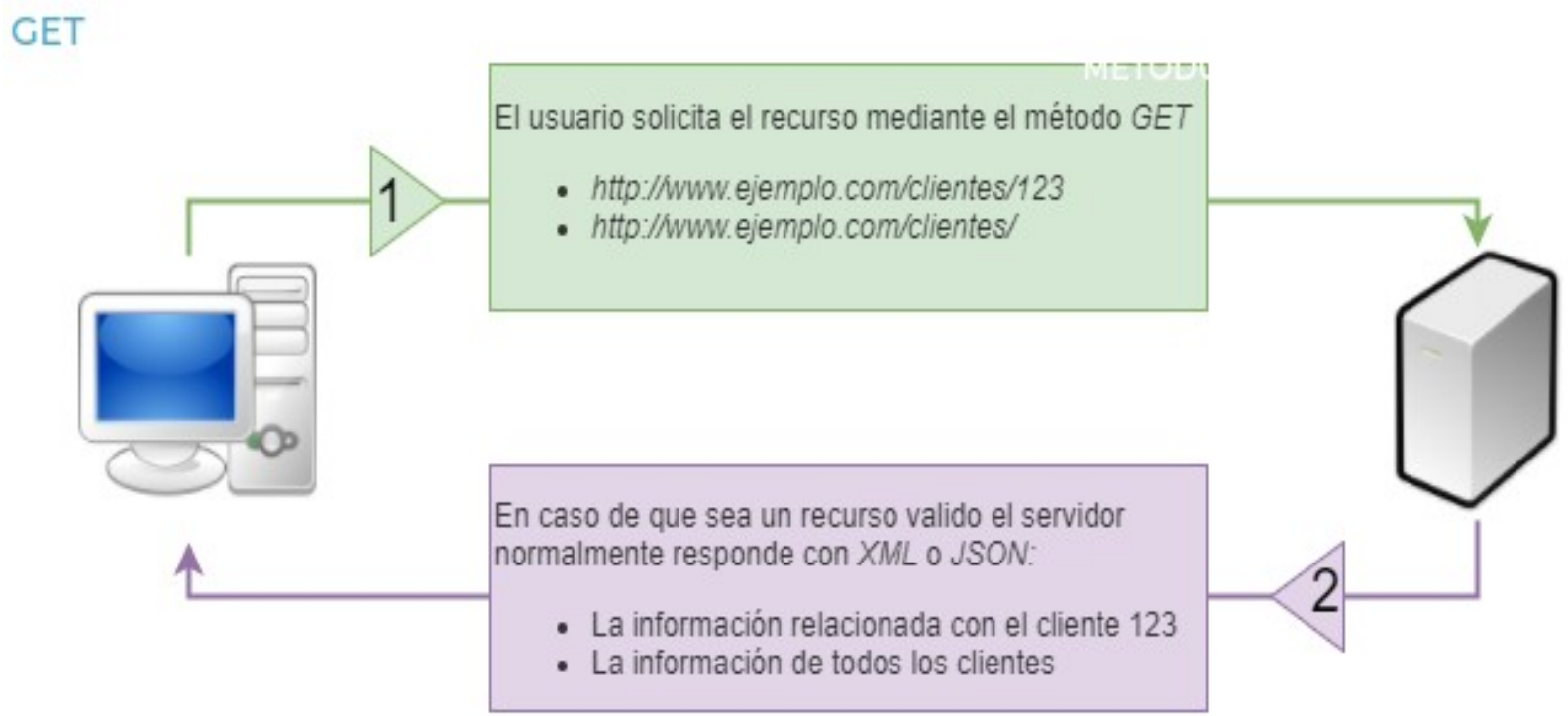
# Servicio Web RESTful

- Los métodos HTTP que usan los Servicios Web RESTful  son los siguientes:
  - Listar y leer: Usan el método GET
  - Crear: Usan el método POST
  - Actualizar: Usan el método PATCH para actualizar y PUT para reemplazar.
  - Borrar: Usan el método DELETE

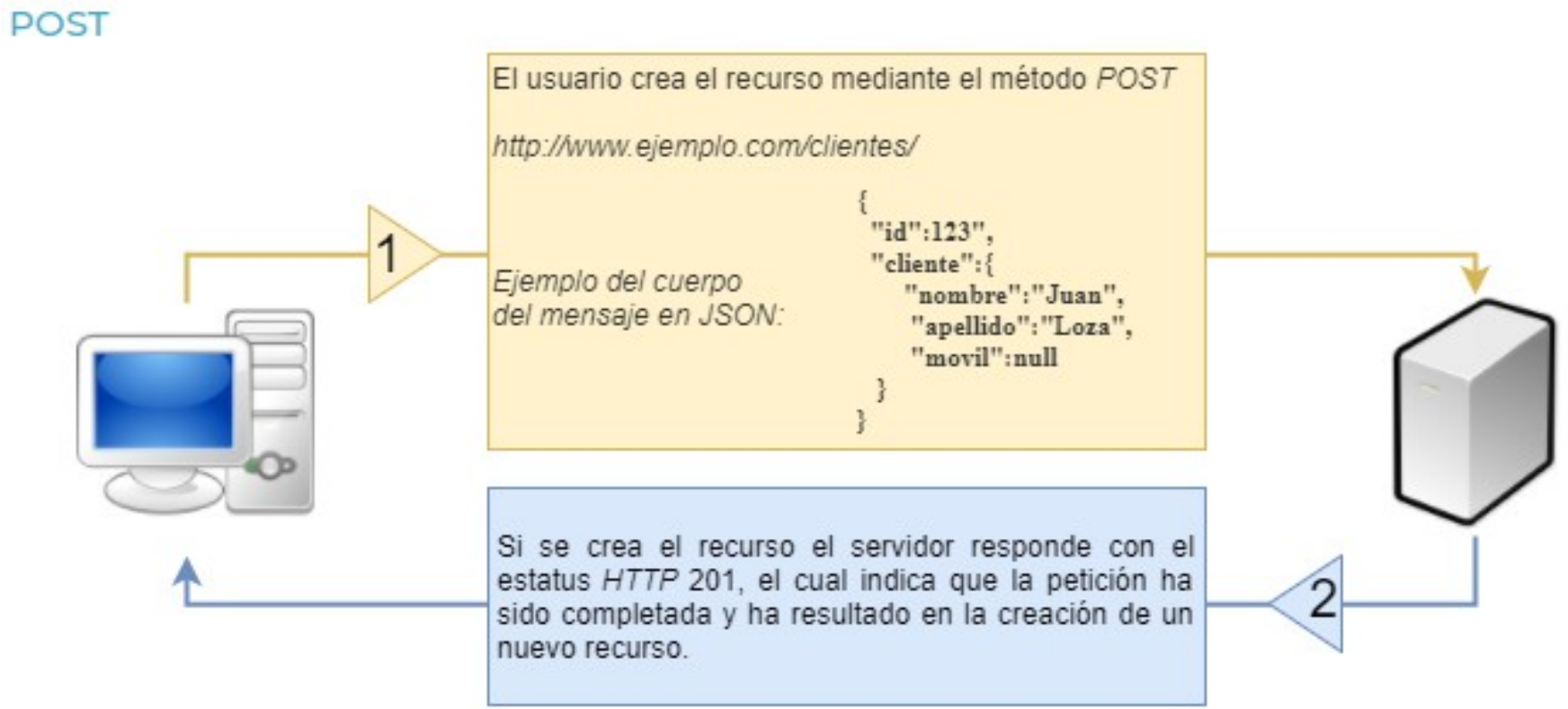
# Servicio Web RESTful

| Operación | Método HTTP | URI | Parámetros | Resultado |
|---|---|---|---|---|
| Listar | GET | /{recurso} | No aplica | Lista del tipo de recurso |
| Crear | POST | /{recurso} | Dentro del cuerpo en el POST | Se crea un nuevo recurso |
| Leer | GET | /{recurso}/{recurso_id} | No aplica | Recurso en función al id |
| Actualizar | PATCH/PUT | /{recurso}/{recurso_id} | Se pasan usando una cadena de consulta | Se actualiza/reemplaza el recurso |
| Borrar | DELETE | /{recurso}/{recurso_id} | No aplica | Se elimina el recurso en función al id |

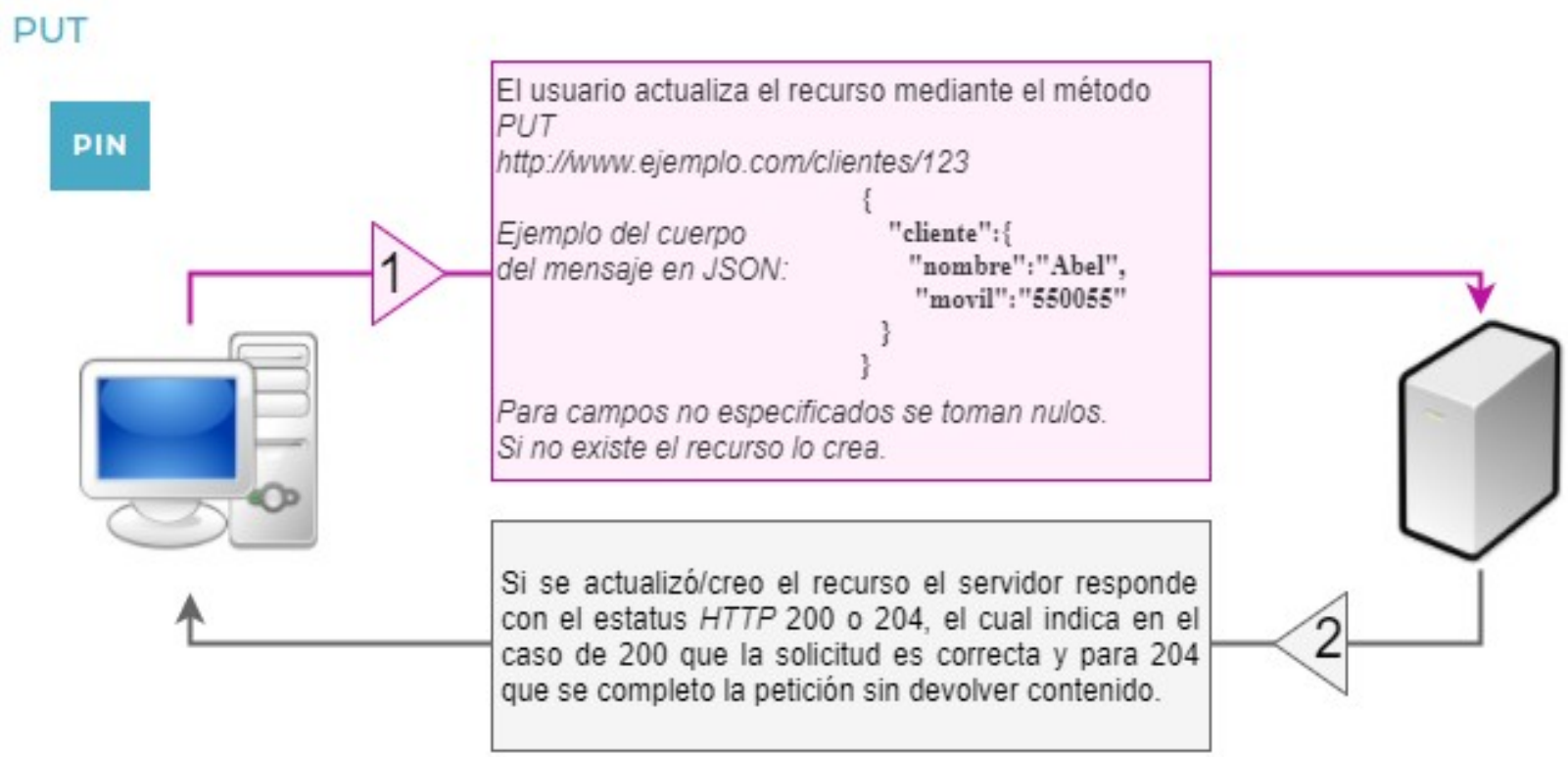
# Servicio Web RESTful



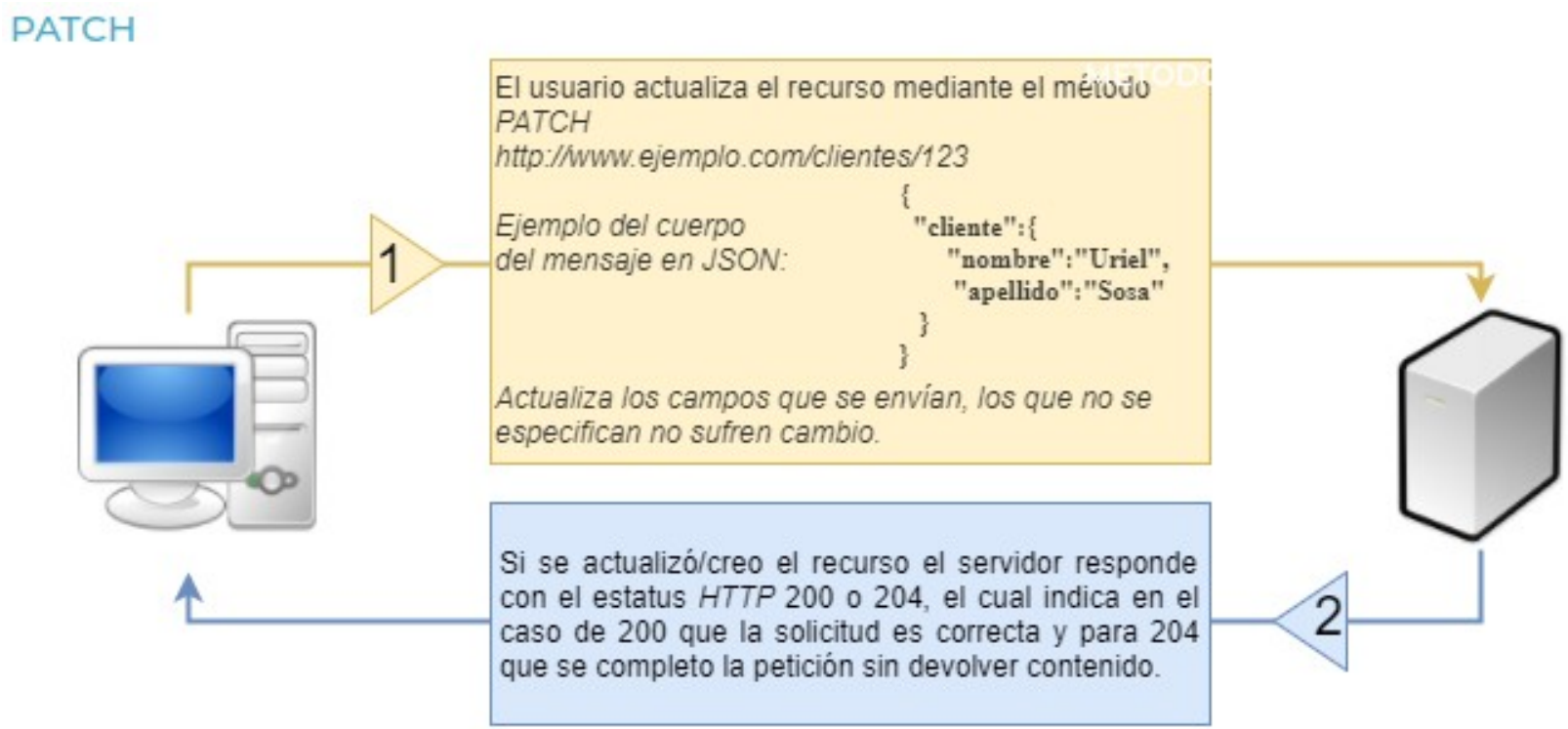
# Servicio Web RESTful



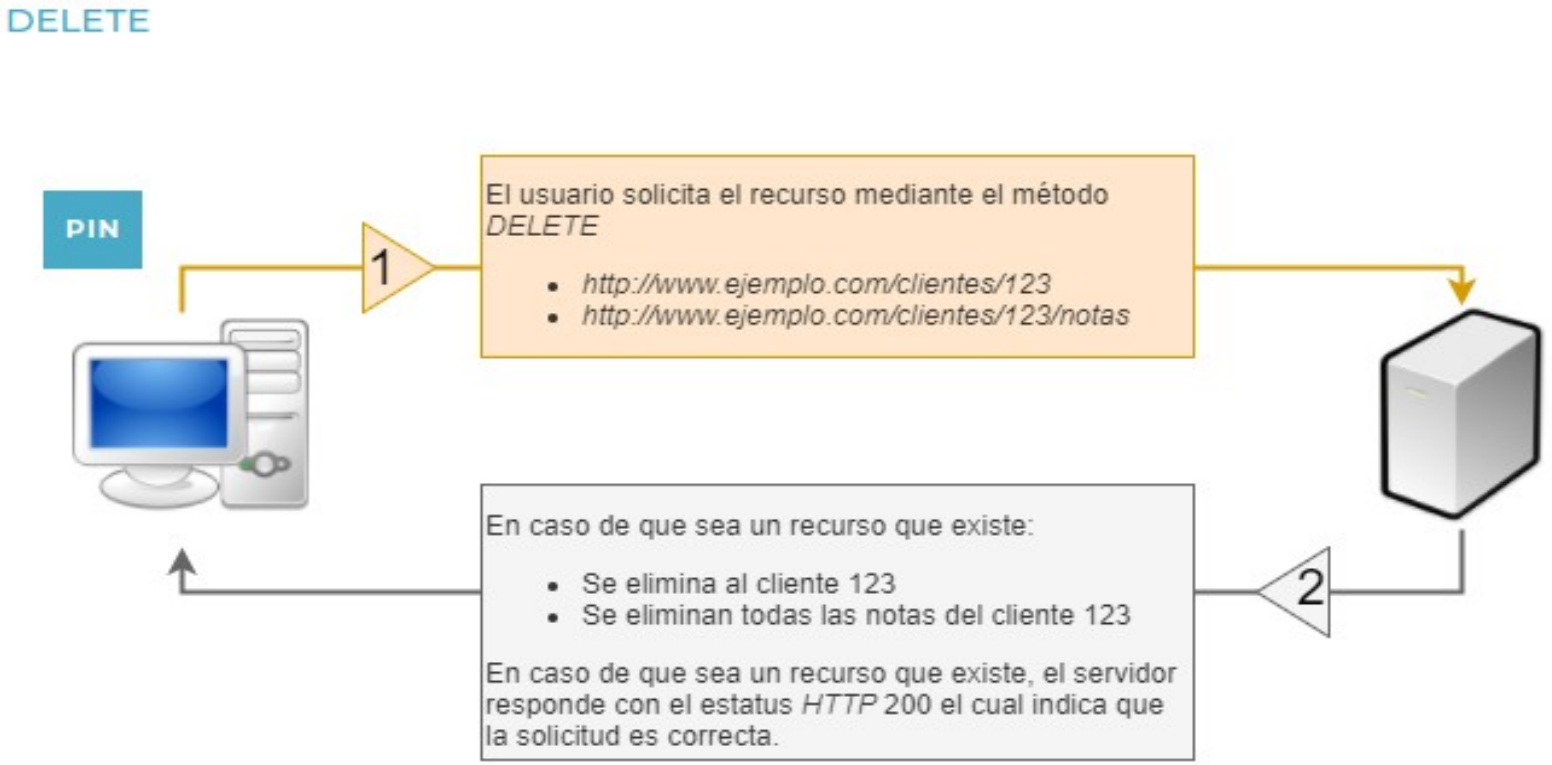
# Servicio Web RESTful



# Servicio Web RESTful



# Servicio Web RESTful



# Jakarta EE

- Jakarta EE es la nueva plataforma de Java Enterprise Edition o Java EE. Hasta este momento todas las especificaciones han sido fuertemente lideradas por Oracle que es el que mantiene la propiedad de Java como lenguaje y por lo tanto de Java EE como extensión natural de este.

# Jakarta EE

- Jakarta EE y Especificaciones:
  - Jakarta RESTFul Web Services: La especificación que hace referencia a los servicios Web de tipo REST.
  - Jakarta Enterprise Java Beans: Uno de los proyectos más importantes que es el uso de EJBS.
  - Jakarta Persistence: proyecto que aborda todo lo relacionado a la capa de persistencia.

# Jakarta EE

- En Jakarta EE fue necesario renombrar los paquetes anteriores ya que Oracle mantiene los derechos:
    - Java y javax
  - Por
    - jakarta

# Jakarta Servicios Web RESTFul

- Jakarta EE contiene la especificación JAX-RS para la implementación de Servicios Web Restful.
- Utiliza anotaciones Java para simplificar el desarrollo de servicios web RESTful, por lo tanto, la reflexión en tiempo de ejecución generará las clases auxiliares y los artefactos para el recurso.
- Un archivo de aplicación Jakarta EE que contenga clases de recursos REST de Jakarta tendrá los recursos configurados, las clases auxiliares y los artefactos generados y el recurso expuesto a los clientes mediante la implementación del archivo en un servidor Jakarta EE.

# Instalación JAX-RS

- Para realizar la instalación del API JAX-RS en un proyecto Java administrado por Maven es necesario contar con las siguientes dependencias:
- <dependency>
  - <groupId>jakarta.platform</groupId>
  - <artifactId>jakarta.jakartaee-web-api</artifactId>
  - <version>9.0.0</version>
- </dependency>
- <dependency>
  - <groupId>jakarta.ws.rs</groupId>
  - <artifactId>jakarta.ws.rs-api</artifactId>
  - <version>3.0.0</version>
- </dependency>

# Anotaciones JAX-RS

- @ApplicationPath
  - La anotación @ApplicationPath se usa para definir la asignación de URL para la aplicación. La ruta especificada por @ApplicationPath es el URI base para todos los URI de recursos especificados por las anotaciones
  - Solo puede aplicar @ApplicationPath a una subclase de jakarta.ws.rs.core.Application

    - @ApplicationPath("webservice")

# Anotaciones JAX-RS

- @Path
  - El valor de la anotación @Path es una ruta URI relativa que indica dónde se alojará la clase Java: por ejemplo, /helloworld.
  - Puede incrustar variables en los URI para crear una plantilla de ruta de URI. Por ejemplo, podría solicitar el nombre de un usuario y pasarlo a la aplicación como una variable en la URI: /helloworld/{username}.

    - @Path(“/helloworld”)

# Anotaciones JAX-RS

- Anotaciones que definen el método HTTP de consumo:
  - @GET
  - @POST
  - @PUT
  - @DELETE
  - @PATCH
  - @HEAD
  - @OPTIONS

# Anotaciones JAX-RS

- @PathParam
  - La anotación @PathParam es un tipo de parámetro que se puede extraer para usar en su clase de recurso. Los parámetros de ruta de URI se extraen del URI de solicitud y los nombres de los parámetros corresponden a los nombres de variables de la plantilla de ruta de URI especificados en la anotación de nivel de clase @Path.

    - @PathParam("id") int id

# Anotaciones JAX-RS

- @QueryParam
  - La anotación @QueryParam es un tipo de parámetro que se puede extraer para usar en su clase de recursos. Los parámetros de consulta se extraen de los parámetros de consulta del URI de solicitud.

    - @QueryParam(“name") String name
    - http://myapi.com/customers?name=oscar

# Anotaciones JAX-RS

- @Consumes
  - La anotación @Consumes se usa para especificar los tipos de medios MIME de representaciones que un recurso puede consumir y que fueron enviados por el cliente.

    - @Consumes(MediaType.APPLICATION_JSON)

# Anotaciones JAX-RS

- @Produces
  - La anotación @Produces se usa para especificar los tipos de representaciones de medios MIME que un recurso puede producir y enviar al cliente: por ejemplo, "texto/simple".

    - @Produces(MediaType.APPLICATION_XML)

# Servicio Web Restful Jakarta EE

- Ejemplo de Configuración:

  – @ApplicationPath("ws")
  – public class WSAppConfig extends Application {}

# Servicio Web Restful Jakarta EE

- Ejemplo de EndPont:

  - @Path("/actions")
  - public class DemoEndPoint {
    - @GET
    - @Path("/hello")
    - public Response hello() {
      - return Response.ok().entity("Service online").build();
      - }
    - }

# Consumo WS Restful

- Para consumir un Servicio Web Restful, es necesario crear un cliente.
- Jakarta EE define las siguientes clases:
  - Client
  - ClientBuilder

# Consumo WS Restful

- Ejemplo de cliente sencillo:

- Client client = ClientBuilder.newClient();
- String name = client.target("http://example.com/webapi/hello") .request(Media Type.TEXT_PLAIN).get(String.class);

# Consumo WS Restful

- WebTarget
  - El destino de un cliente, el recurso REST en un URI particular, está representado por una instancia de la interfaz jakarta.ws.rs.client.WebTarget. Obtiene una instancia de WebTarget llamando al método Client.target y pasando el URI del recurso REST de destino.
    - Client client = ClientBuilder.newClient();
      - WebTarget myResource = client.target("http://example.com/webapi");
    - WebTarget myResource = client.target(REST_URI_JSON_GET)
      .path("{id}").resolveTemplate("id", 3);

# Consumo WS Restful

- Si el recurso REST de destino está esperando una solicitud HTTP POST, llame al método Invocación.Builder.post.

  – Client client = ClientBuilder.newClient();
  – StoreOrder order = new StoreOrder(...);
  – WebTarget myResource = client.target("http://example.com/webapi/write");
  – TrackingNumber trackingNumber = myResource.request(MediaType.APPLICATION_XML)
  .post(Entity.xml(order), TrackingNumber.class);

# Consumo WS Restful

- Otro ejemplo:

- Client client = ClientBuilder.newClient();
- WebTarget myResource = client.target(REST_URI_JSON_GET)
  .path("{id}").resolveTemplate("id", 3);
- Response resp =
  myResource.request(MediaType.APPLICATION_JSON).get();
  DataPojo data = resp.readEntity(DataPojo.class);

# Consumo WS Restful

- DataPojo data = new DataPojo(8, "information");
- Client client = ClientBuilder.newClient();
- WebTarget myResource = client.target(REST_URI_JSON_POST);
- DataPojo dataResp = myResource.request(MediaType.APPLICATION_JSON) .post(Entity.json(data), DataPojo.class);

# Contacto

Jorge Alberto Montalvo Olvera

*Ingeniero en Computación*

jorge.Montalvo@gm3s.com.mx