

## TEMA 6. PROGRAMACIÓN DE COMPONENTES DE ACCESO A DATOS.

### Objetivos

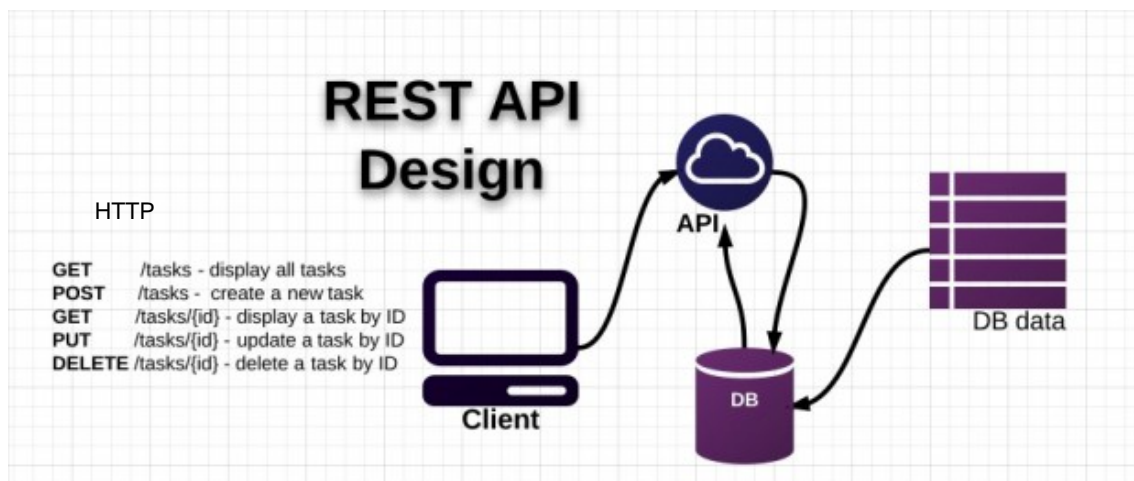
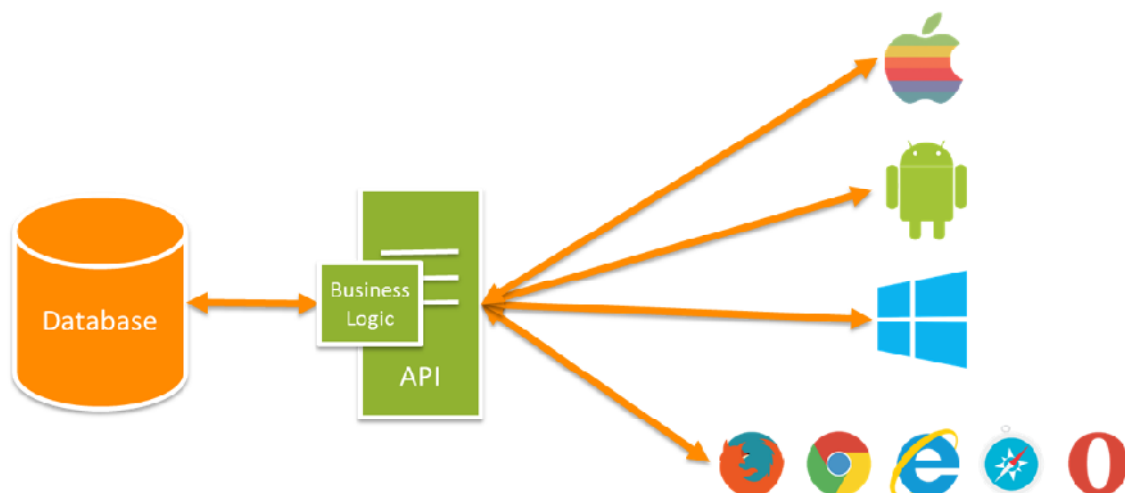
- Conocer que es una API
- Desarrolla una API con conexión a base de datos
- Probar API desde Postman
- Consumir API desde Java

### Contenidos

#### 1.- ¿Qué es una API?

API son las siglas de **Application** Programming Interface, o Interfaz de Programación de Aplicaciones. Es la forma en que los programas y/o los sitios webs **intercambian datos**.

Ofrecer y/o consumir datos desde varias aplicaciones, independientemente de la plataforma. El formato de intercambio de datos normalmente es JSON o XML.



Tipos de API:

~~SOAP~~: Simple Object Access Protocol.

- Comunicación siempre mediante XML.
- Implementa protocolos de seguridad.
- Más pesado: Agrega metadatos (cabecera y otra info)

Anticuoado

**REST**: Representational State Transfer.

- Más ligero: Json mejor que XML. Muy ligero, sus respuestas contienen exactamente la información que se necesita (datos planos), pues utiliza pocos recursos.
- Arquitectura sin estado, significa que cada petición al servidor es tratada de manera totalmente independiente. La ventaja está en la simplicidad del componente servidor y lo fácil que resulta escalar y hacer crecer el sistema.
- Es flexible para sus respuestas. Pueden ser xml o json.
- Es fácil y simple de interpretar.

Para hacer las peticiones al servidor se realizan mediante una **URI** (Identificador de Recursos Uniforme).

Con esto nos podemos comunicar

Sintaxis de una URI:

- Evitar acciones y verbos, siempre nombrar recursos.
- Recursos siempre en plural
- Separación de palabras en los recursos con - o \_

<b>{protocolo}: //{dominio o hostname}:{puerto}/{ruta recurso}/id</b>
---

Ejemplo:

*http://90.100.23.30:8888/facturas/234/editar* ❌

*http://90.100.23.30:8888/factura/234* ❌

*http://90.100.23.30:8888/Facturas/234* ❌

**PUT:** *http://90.100.23.30:8888/facturas/234* ✅

Usar verbos HTTP para realizar acciones sobre el recurso:

- GET: Para **consultar** <sup>como listar</sup> recursos. Se puede indicar un identificador para consultar uno en concreto.  
*/facturas{/id}*
- POST: Para **crear** nuevos recursos.  
*/facturas*
- PUT: Para **editar** recursos. Se debe indicar el identificador del recurso.  
*/facturas/id*
- DELETE: Para **eliminar** recursos.  
*/facturas/id*

Ejemplo:

GET → <http://localhost:8888/ApiHolaMundo/facturas>  
GET → <http://localhost:8888/ApiHolaMundo/facturas/5>  
POST → <http://localhost:8888/ApiHolaMundo/facturas>  
PUT → <http://localhost:8888/ApiHolaMundo/facturas/7>  
DELETE → <http://localhost:8888/ApiHolaMundo/facturas/6>

Resumiendo tenemos que un Servicio Web es un conjunto de protocolos que permiten el intercambio de información entre diferentes aplicaciones.

Solicitamos la página por HTTP y obtenemos la respuesta. Una Solicitud a un servicio web REST nos comunicamos por HTTP y obtenemos la respuesta en formato json por HTTP.

Los recursos REST se basan en URI, por ejemplo: <sup>Servicio que solicitamos</sup>

URL <a href="http://www.libreia.com/libros/293">www.libreia.com/libros/293</a> Código del libro
---

Los métodos http son GET (Consulta), POST (Inserta), PUT (Actualiza) y DELETE (Eliminar).  
Ejemplo:

GET	/libros	<b>Lista todos los libros</b>	Devuelve en bruto con un JSON con clave valor
GET	/libros/12	<b>Consulta el libro de id 12</b>	Este también devuelve JSON
POST	/libros	<b>Crea un nuevo libro</b>	Necesita la estructura JSON
PUT	/libros/12	<b>Actualiza el libro con id 12</b>	Necesita los nuevos datos que quiere actualizar
DELETE	/libros/12	<b>Elimina el libro con id 12</b>	No devuelve nada y no hace falta mandarle nada

Los Metadatos son los códigos de estado que devolverá el Servicio Web REST:

<b>200</b>	<b>Exito</b>	Todo lo que este por encima de este número, estará bien
<b>300</b>	<b>Redirecciones</b>	
<b>400</b>	<b>Errores Cliente</b>	
<b>500</b>	<b>Errores Servidor</b>	

Los tipos de contenido que enviamos o recibimos son del tipo

- text/plain
- text/xml
- application/json (más habitual)

## 2.- Crear API REST desde Java.

Para poder crear una API RestFul con acceso a base de datos necesitamos los siguientes requerimientos técnicos:

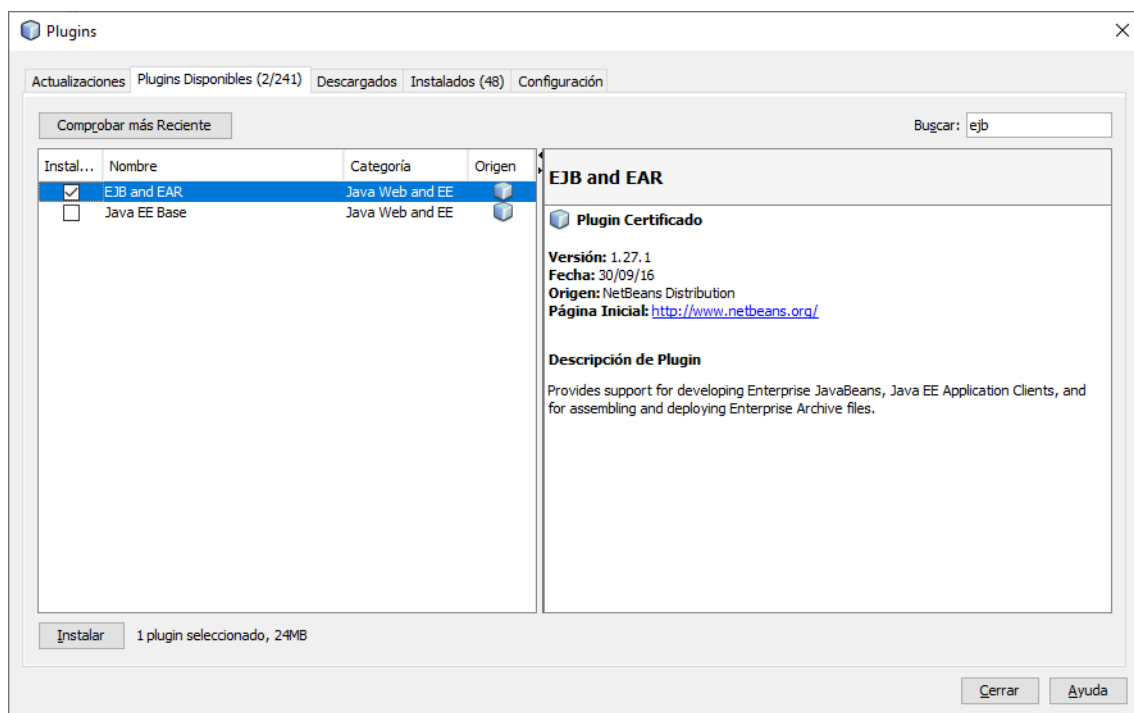
JDK Java.

Netbeans 8.2 con Java EE para poder crear proyectos Java Web.

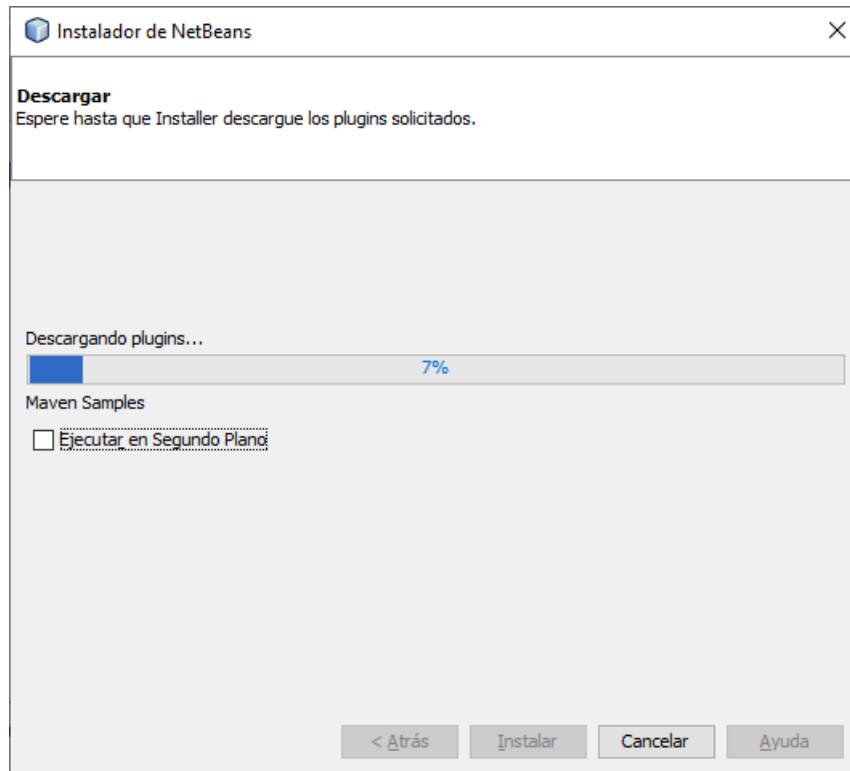
Servidor GlassFish.

### ~~Instalar Java EE en Netbeans.~~

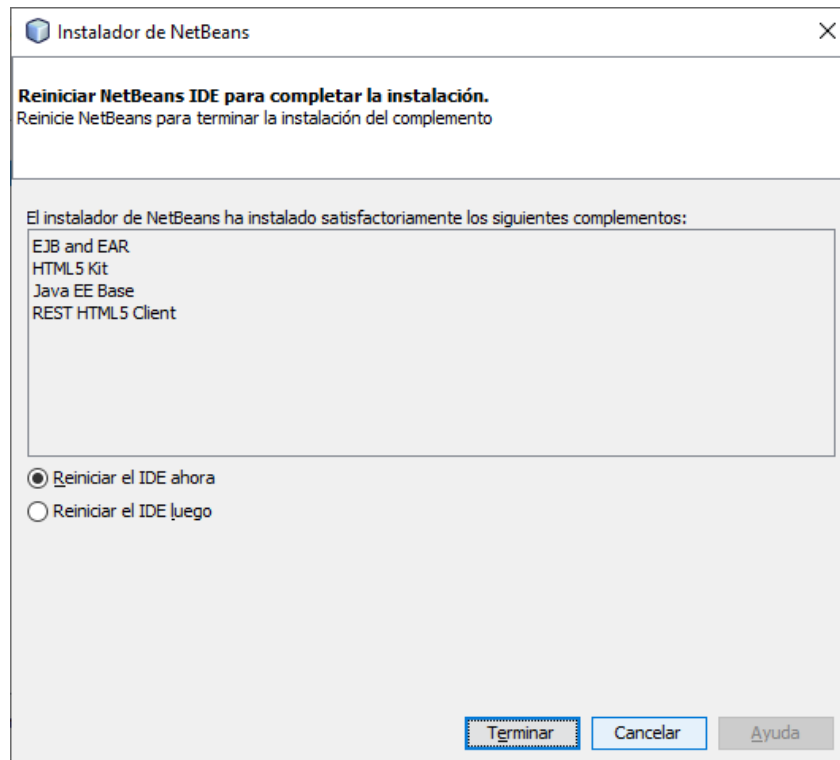
Si no tenemos instalado Java EE en Netbeans podemos instalarlo Herramientas/Plugins/ Plugins Disponibles. Buscar "EJB and Ear", seleccionar e Instalar.



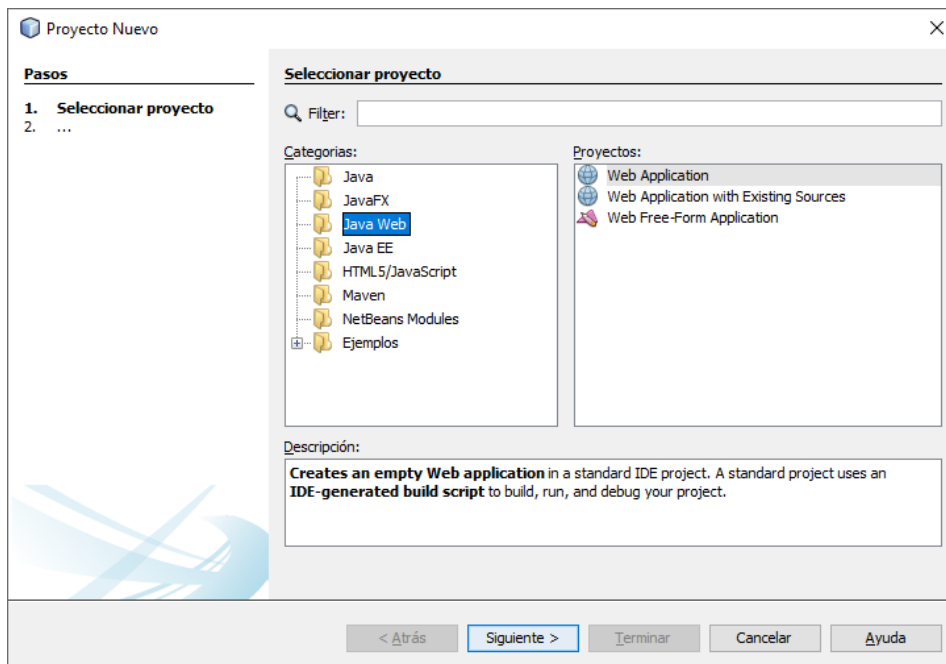
Marcamos que aceptamos la licencia e instalamos.



Tras unos minutos JavaEE queda instalado. Reiniciamos el IDE.



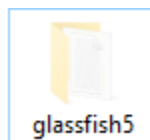
Pulsamos Nuevo/Proyecto y se muestra la opción de crear proyectos Java Web.



### ~~Instalar Servidor GlassFish.~~

Glassfish es un servidor de aplicaciones que implementa la plataforma JavaEE5. Podemos descargar el Servidor GlassFish desde [glassfish.java.net](http://glassfish.java.net) – download.

Descargamos la version 5 (Full). Descomprimir en la raíz del disco C:\ y ejecutamos la consola cmd.



Accedemos a la carpeta `cd c:\glassfish5\bin` y ejecutar **asadmin change-admin-password** para cambiar password del admin. Introducimos el usuario **admin** sin password (**intro**) e introducimos y repetimos el nuevo password (por ejemplo **1111**).

```

Simbolo del sistema
c:\glassfish5\bin>cd c:\glassfish5\bin

c:\glassfish5\bin>asadmin change-admin-password
Enter admin user name [default: admin]>
Enter the admin password>
Enter the new admin password>
Enter the new admin password again>
Command change-admin-password executed successfully.

c:\glassfish5\bin>

```

Si tenemos problemas al cambiar password de glassfish puede ser por la versión del jdk. Para solucionarlo debemos abrir el fichero **asenv.bat** de la ruta **C:\glassfish5\glassfish\config** y al

final del fichero añadir la línea siguiente, siendo necesario especificar la ruta de la versión de JDK que tenemos instalada:

```
set AS_JAVA=C:\Program Files\Java\jdk1.8.0_111
```

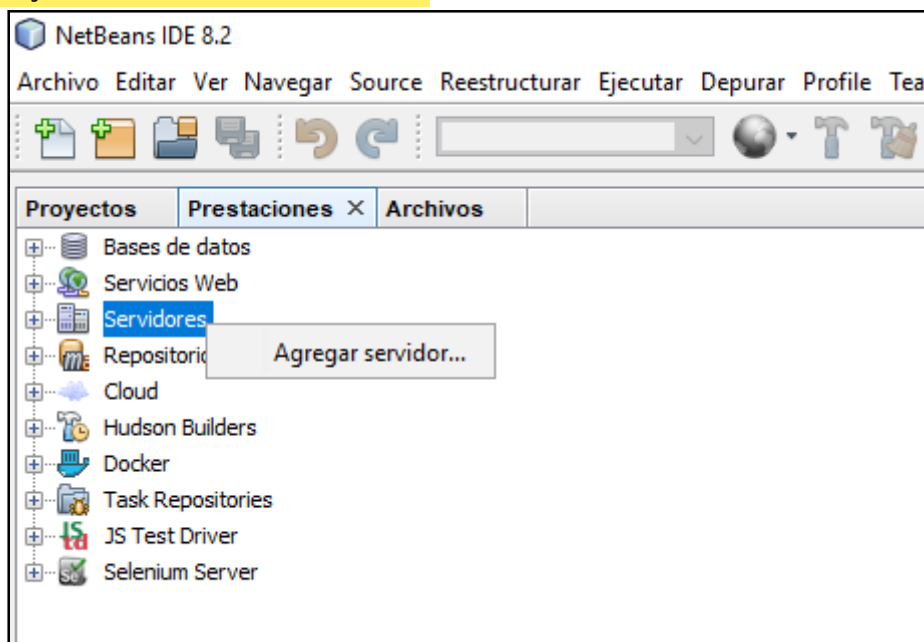
A continuación ejecutar **asadmin start-domain domain1** para poner en marcha el servidor glassfish.

```
Símbolo del sistema

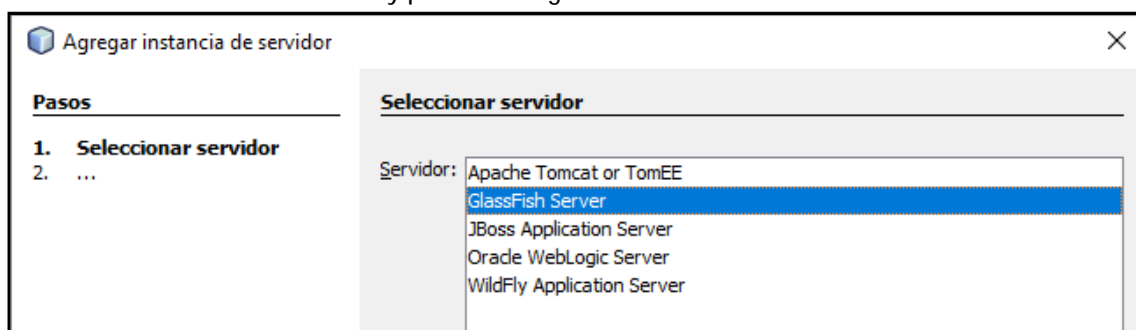
c:\glassfish5\bin>asadmin start-domain domain1
Waiting for domain1 to start .....
Successfully started the domain : domain1
domain Location: C:\glassfish5\glassfish\domains\domain1
Log File: C:\glassfish5\glassfish\domains\domain1\logs\server.log
Admin Port: 4848
Command start-domain executed successfully.

c:\glassfish5\bin>
```

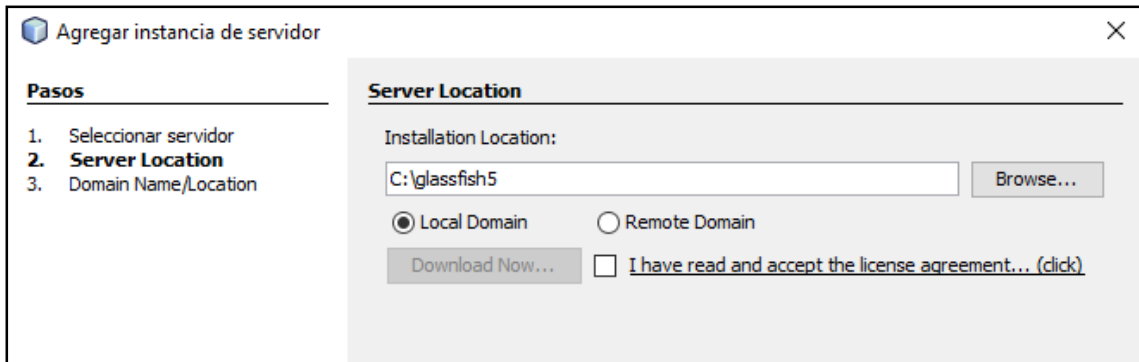
Ejecutamos Netbeans y en la pestaña **Services(Prestaciones)** pulsamos botón derecho sobre Servidores y seleccionamos **Añadir Servidor**.



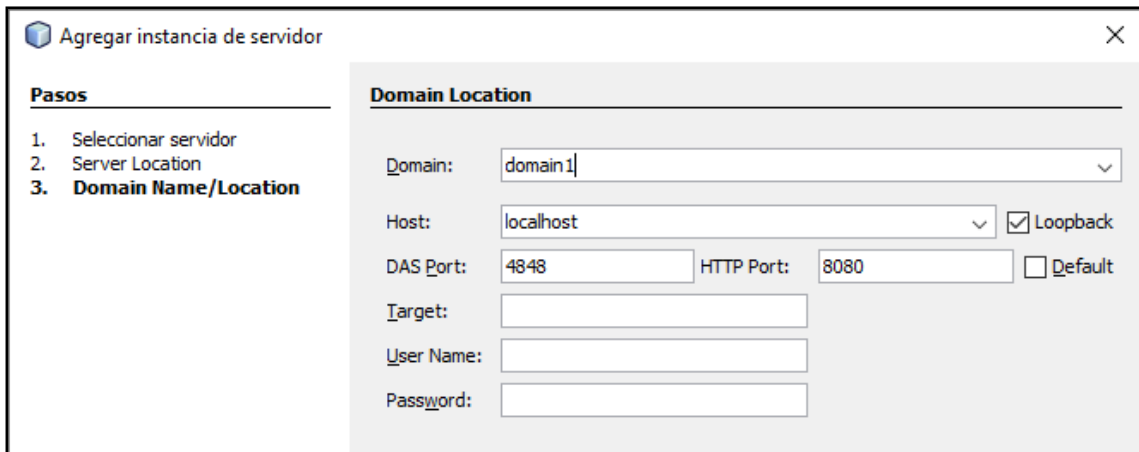
Seleccionamos GlassFish Server y pulsamos Siguiente.



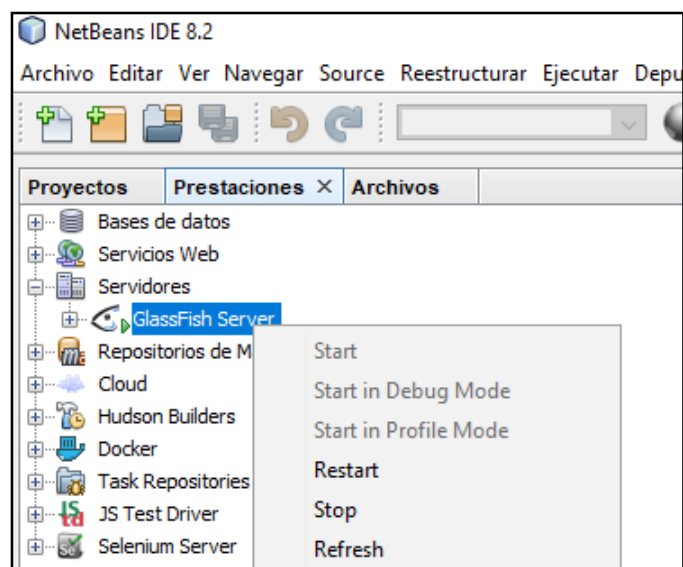
Seleccionamos la carpeta del Servidor **C:\glassfish5** y pulsamos Siguiente.



Introducimos el **usuario y password** del administrador del servidor (usuario: admin y pass: 1111) y pulsamos **Terminar**.



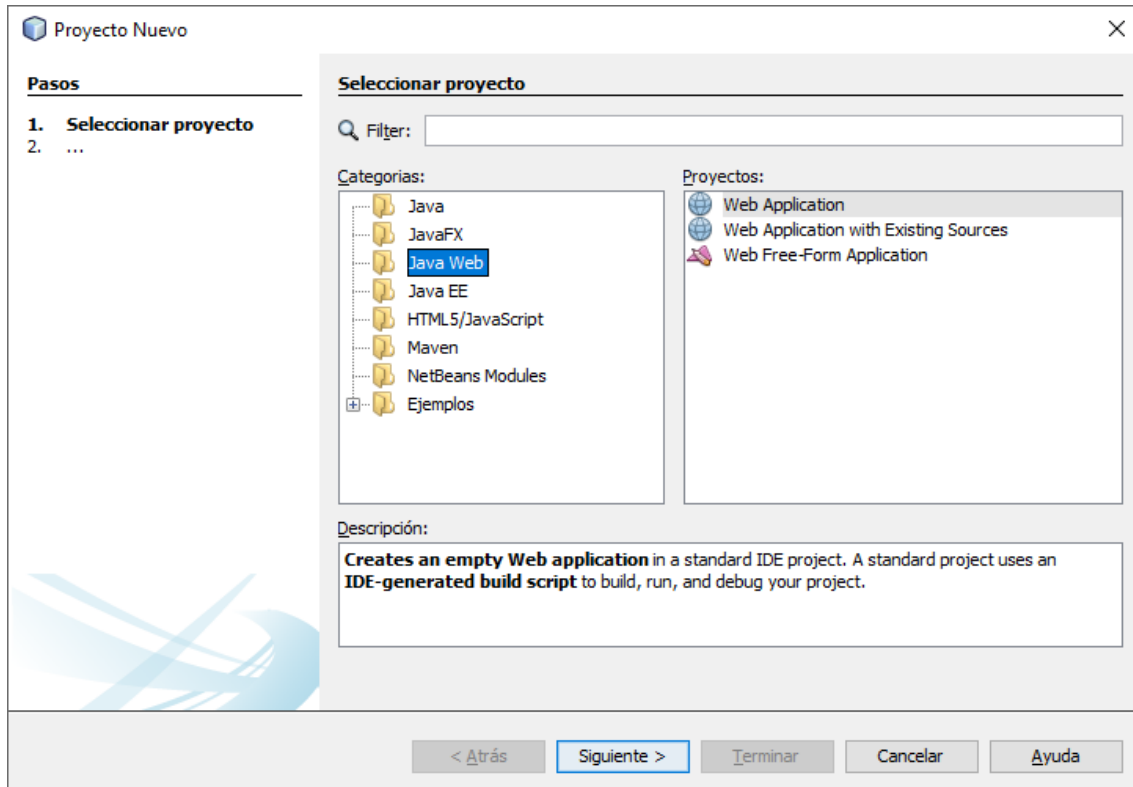
Muestra el GlassFish Server en Servidores. Aparece en marcha y podemos detener, pausar y arrancar con el botón derecho.



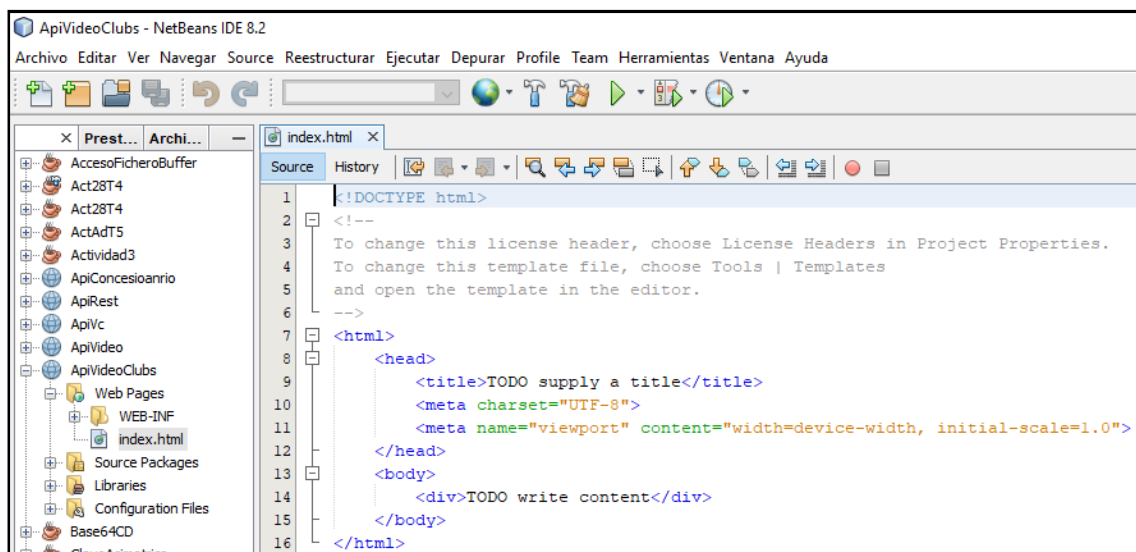


## Proyecto Java Web.

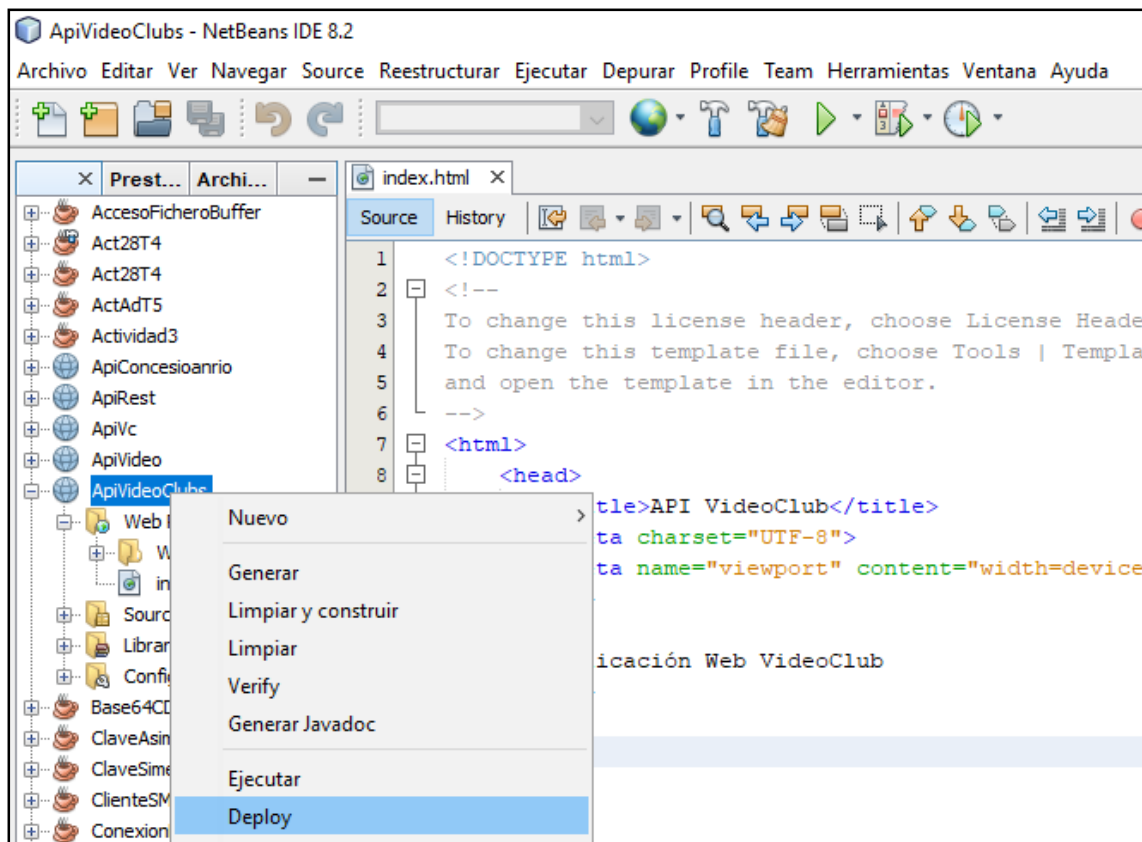
Creamos nuevo proyecto desde Archivo / Proyecto Nuevo y seleccionamos Java Web/Web Application.



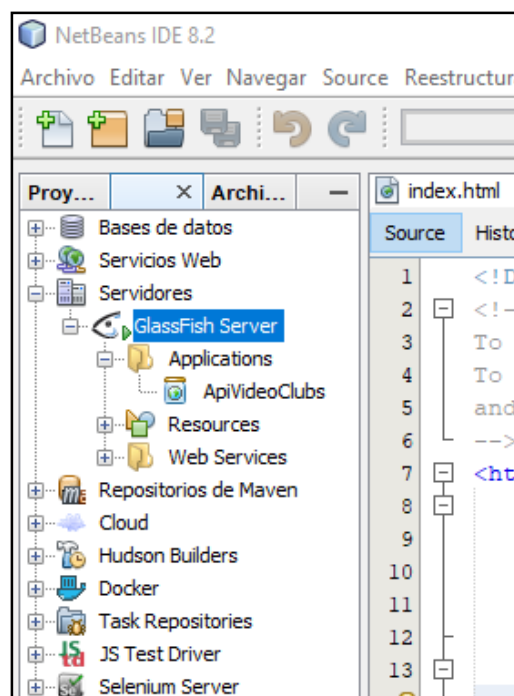
Pulsamos Siguiete y damos nombre al proyecto (por ejemplo **ApiVideoClubs**). Seleccionamos el Servidor Glassfish y Terminar. Se crea un proyecto web y se abre la página principal del proyecto **index.html**



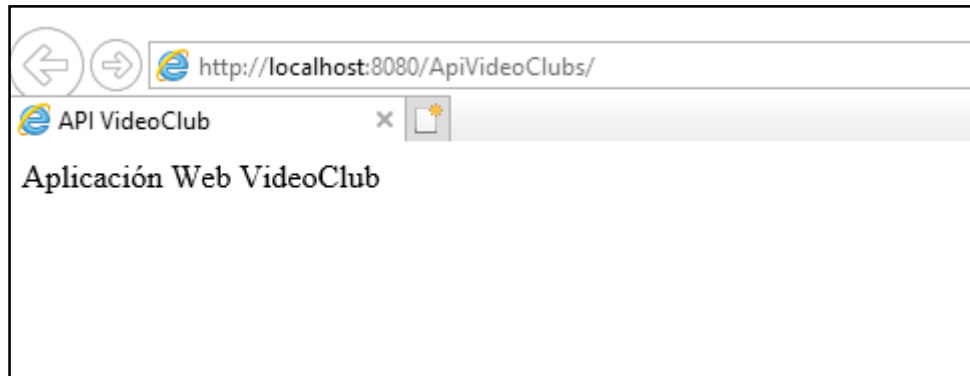
Podemos cambiar el texto que aparece en el <title> y el <body> de la web. Por ejemplo **API Videoclub** y **Aplicación Web VideoClub**. Pulsamos botón derecho sobre el proyecto y **Deploy**.



Accedemos a Prestaciones(Services) y desplegamos el Servidor Web Glassfish. Desplegamos Applications y vemos que aparece el proyecto Web recientemente creado.



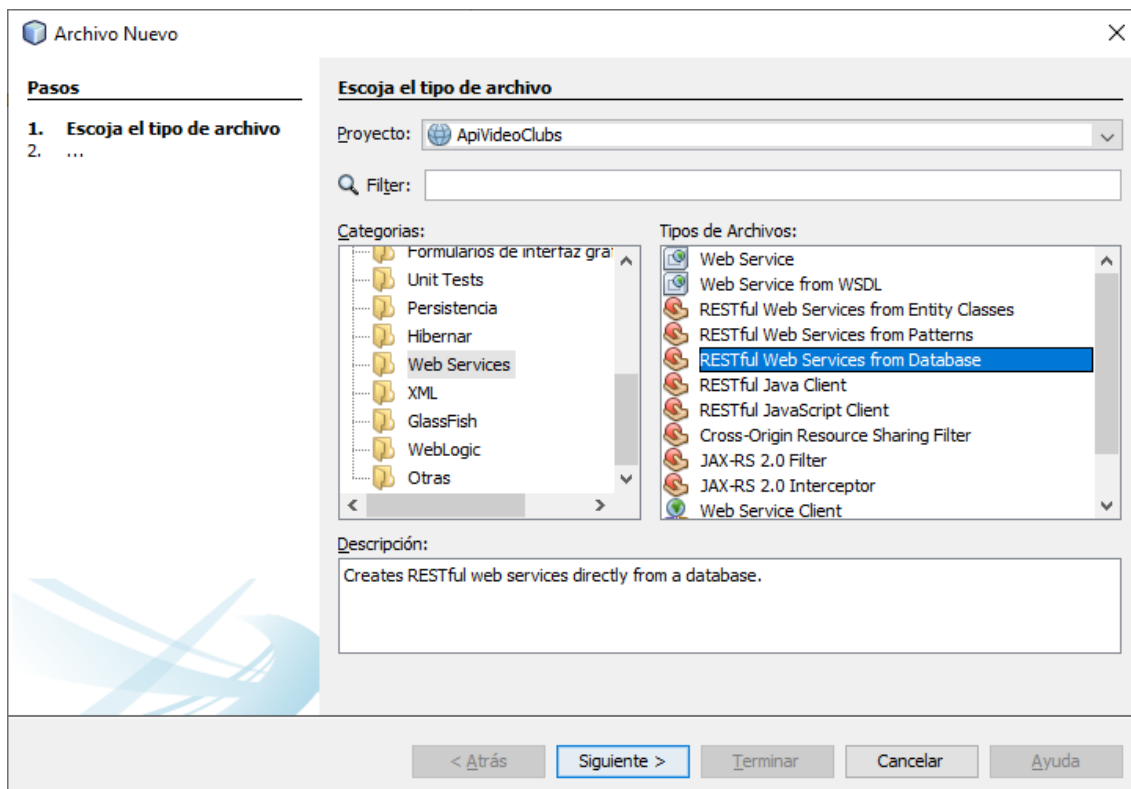
Pulsamos botón derecho sobre el proyecto de Glassfish y **Open in Browser**. Se nos abre en el navegador con la URL **http://localhost:8080/ApiVideoClubs**. Utiliza el puerto del servidor 8080 y accedemos a la carpeta del proyecto ApiVideoClubs mostrando el fichero **index.html**. Desde cualquier navegador podemos acceder a la web mediante dicha URL.



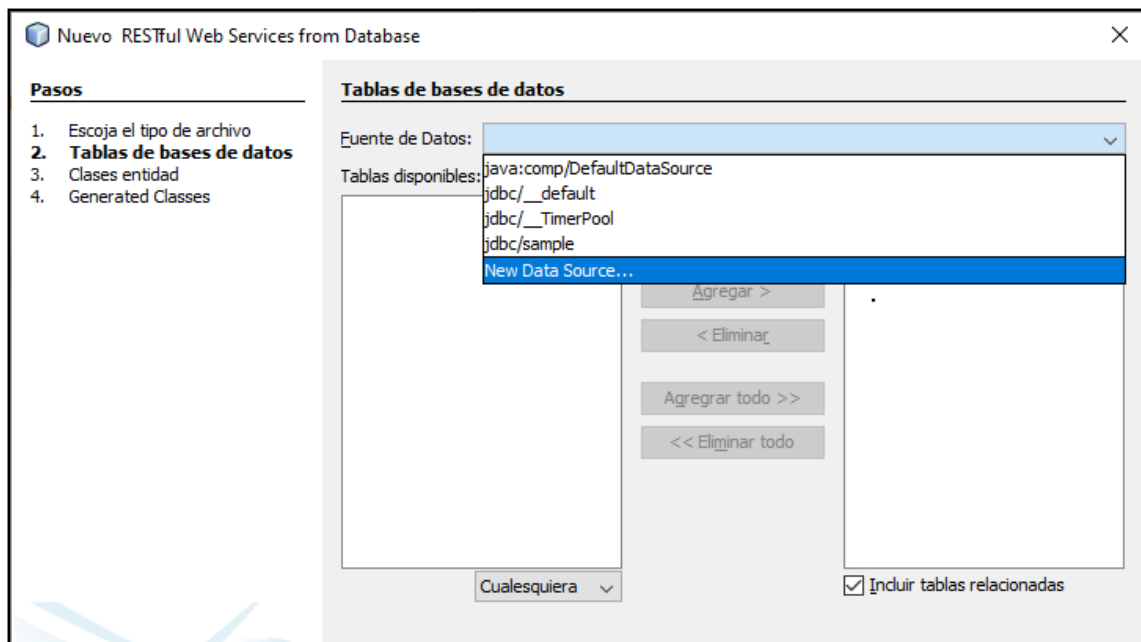
### Crear API RestFul.

Vamos a crear una API RestFul con acceso a base de datos MySQL por lo que debemos tener activo el Servidor MySQL de XAMPP.

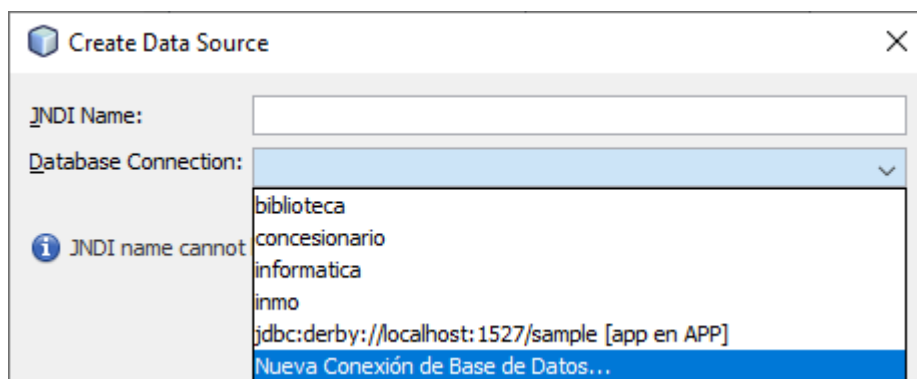
Para crear en el proyecto una API RestFul con acceso a Base de Datos pulsamos botón derecho sobre el proyecto, **Nuevo / Otro** y en la categoría **Web Services** seleccionamos **RestFul Web Services from Database**.



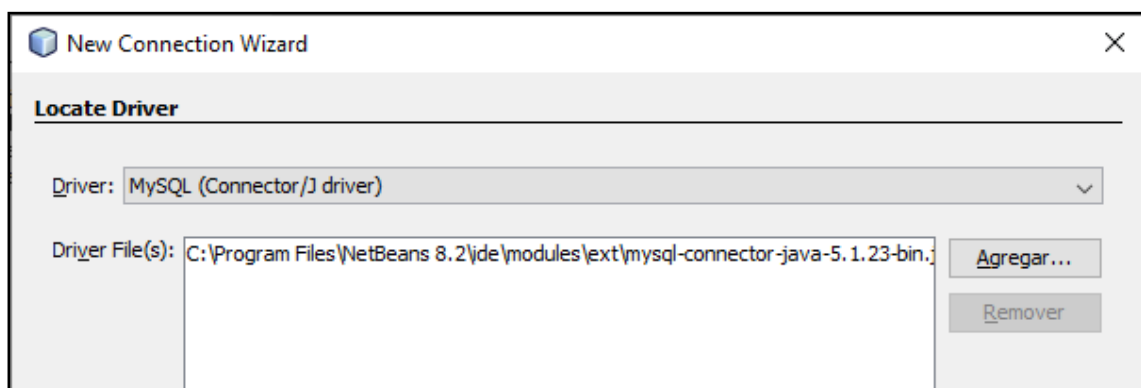
Pulsamos Siguiente y creamos la conexión a la base de datos (por ejemplo videoclub).  
Seleccionamos en la fuente de datos New Data Source...



Seleccionamos **Nueva Conexión de Base de Datos**.



Seleccionamos **MySQL (Connector/J driver)**.



Pulsamos Siguiente y probamos la conexión a la base de datos videoclub con el usuario root y la contraseña vacía.

The screenshot shows the 'New Connection Wizard' dialog box, specifically the 'Customize Connection' step. The dialog has a title bar with a close button. The main area contains several fields and buttons. At the top, there's a dropdown menu for 'Nombre controlador:' set to 'MySQL (controlador Connector/J) on MySQL (Connector/J driver)'. Below it are fields for 'Servidor:' (localhost) and 'Puerto:' (3306). The 'Base de Datos:' field is set to 'videoclub'. The 'Nombre de usuario:' field is set to 'root', and the 'Contraseña:' field is empty. There is a checkbox for 'Recordar contraseña' which is unchecked. Below these fields are two buttons: 'Connection Properties' and 'Probar Conexión'. At the bottom, there's a text field for 'URL de JDBC:' containing 'jdbc:mysql://localhost:3306/videoclub?zeroDateTimeBehavior=convertToNull'. A status bar at the bottom left shows an information icon and the text 'Connection Succeeded.'. At the very bottom, there are five buttons: '< Atrás', 'Siguiente >', 'Terminar', 'Cancelar', and 'Ayuda'. The '< Atrás' button is highlighted with a blue border.

**New Connection Wizard**

**Customize Connection**

Nombre controlador: MySQL (controlador Connector/J) on MySQL (Connector/J driver)

Servidor: localhost Puerto: 3306

Base de Datos: videoclub

Nombre de usuario: root

Contraseña:

☐ Recordar contraseña

Connection Properties Probar Conexión

URL de JDBC: jdbc:mysql://localhost:3306/videoclub?zeroDateTimeBehavior=convertToNull

Connection Succeeded.

< Atrás Siguiente > Terminar Cancelar Ayuda

Pulsamos Siguiente / Siguiente y damos nombre a la conexión. Pulsamos Terminar.

The screenshot shows the 'New Connection Wizard' dialog box, specifically the 'Choose name for connection' step. The dialog has a title bar with a close button. The main area contains a text box with the instruction 'Override the default name for the connection. The name should be descriptive about the connection you are creating.' Below this is a label 'Input connection name:' followed by a text field containing 'videoclub'. At the bottom, there are five buttons: '< Atrás', 'Siguiente >', 'Terminar', 'Cancelar', and 'Ayuda'. The 'Terminar' button is highlighted with a blue border.

**New Connection Wizard**

**Choose name for connection**

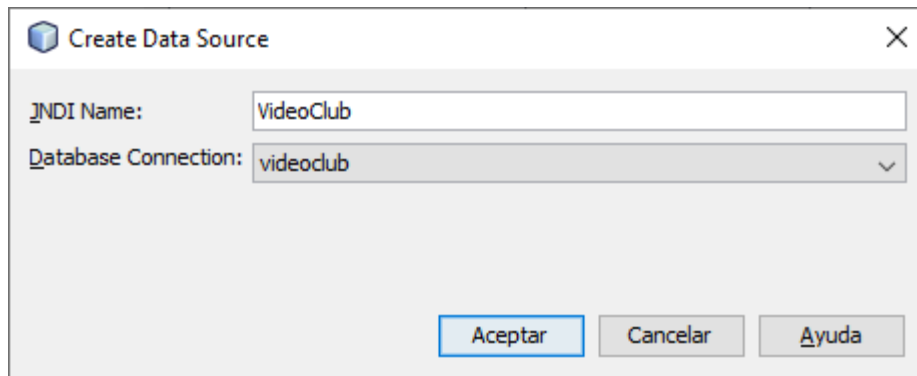
Override the default name for the connection. The name should be descriptive about the connection you are creating.

Input connection name:

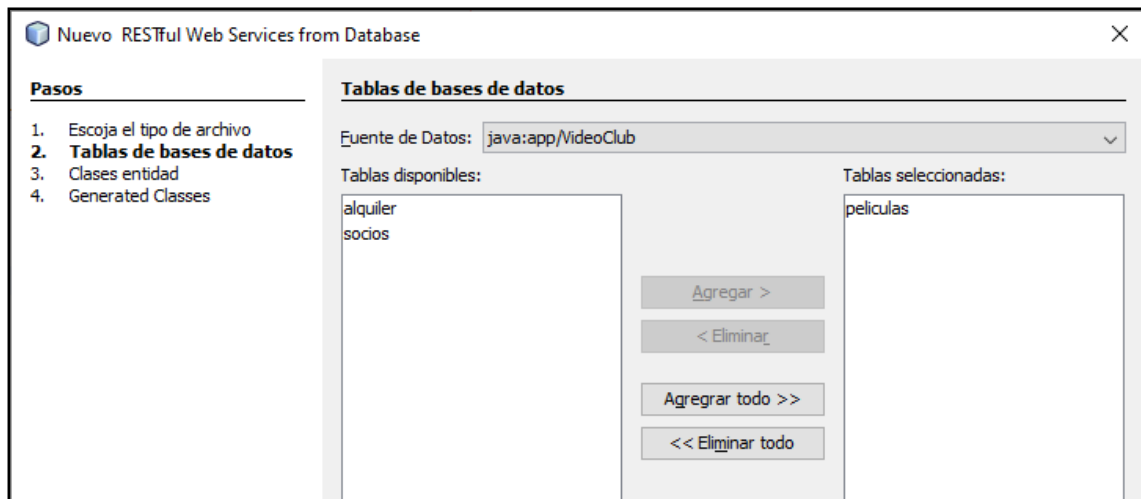
videoclub

< Atrás Siguiente > Terminar Cancelar Ayuda

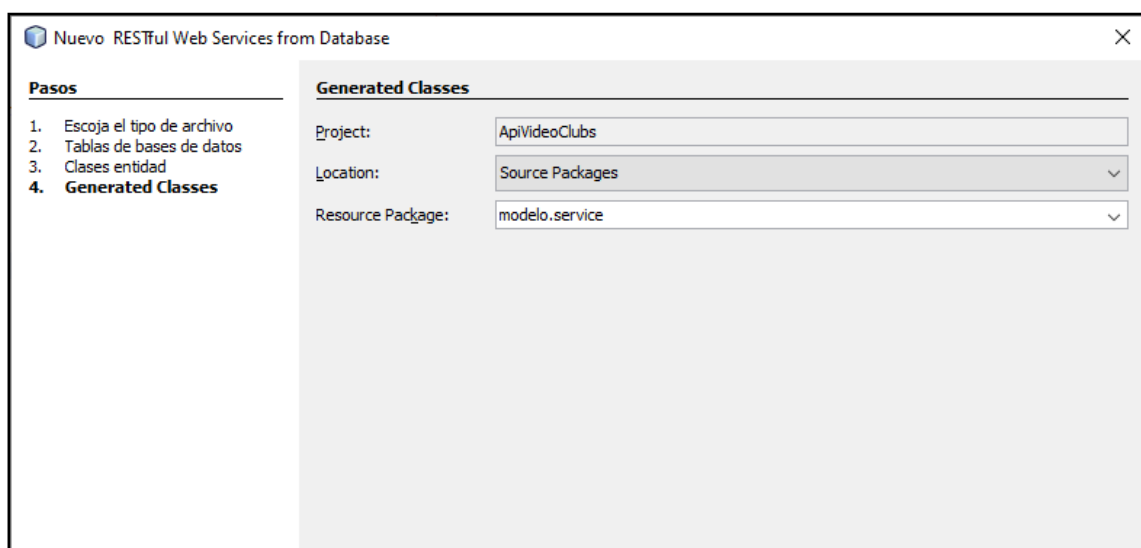
Damos nombre JNDI (por ejemplo VideoClub) y Aceptar.



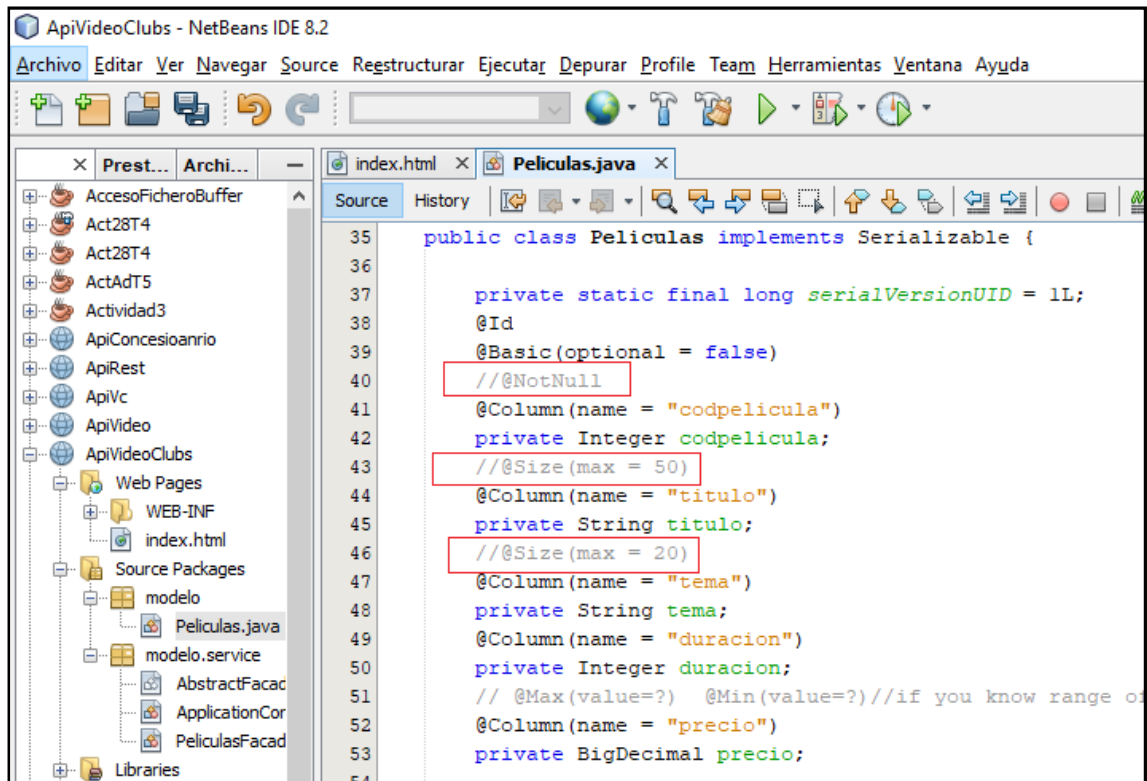
Agregamos la tabla de películas pues vamos a crear una API para dicha tabla.



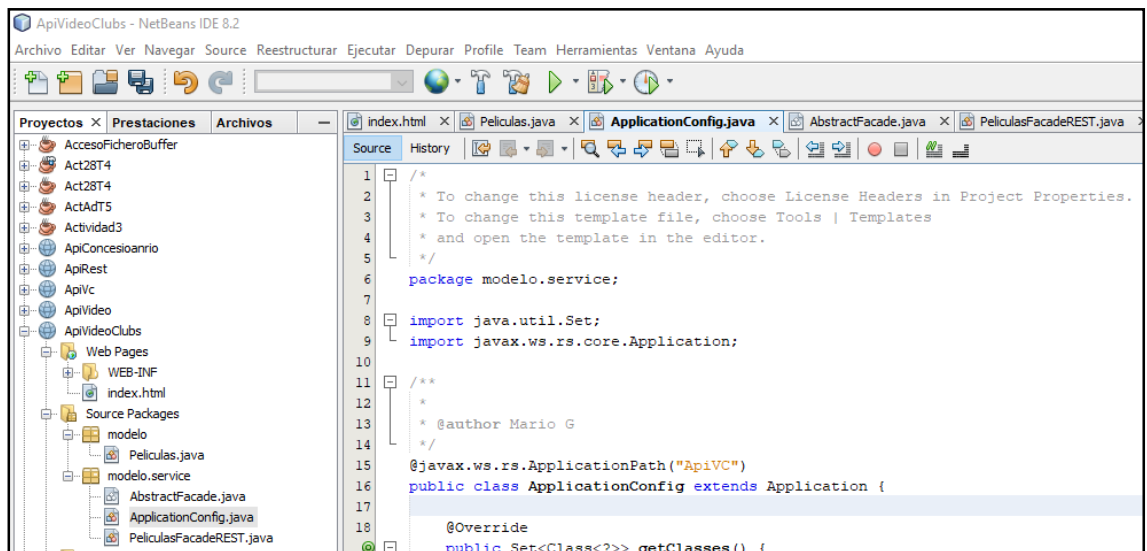
Pulsamos Siguiente y damos nombre al Paquete donde se creará la clase Películas con los atributos y métodos de los campos de la tabla películas, por ejemplo **modelo**. Pulsamos Siguiente y Terminar.



Se crea la clase Peliculas en el paquete modelo. Abrimos la clase Peliculas y comentamos con `//` de los atributos que hacen referencia a los campos de la tabla todas las líneas con anotaciones `@NotNull` y `@Size` y guardamos.



Abrimos el fichero ApplicationConfig.java y cambiamos el nombre del recurso al que accederemos mediante URL (por ejemplo ApiVC).



Abrimos el fichero PeliculasFacadeREST.java y cambiamos el `@Path` de acceso a los métodos (por ejemplo `películas`).

```

23  /**
24   *
25   * @author Mario G
26   */
27  @Stateless
28  @Path("peliculas")
29  public class PeliculasFacadeREST extends AbstractFacade<Pel
30

```

Comprobamos que este fichero desarrolla las peticiones de los métodos GET, POST, PUT y DELETE indicando que datos consume (@Consumes) y que datos produce (@Produces), así como el formato en el que consume y produce la información. Por defecto este formato es XML y JSON (**MediaType.APPLICATION\_XML**, **MediaType.APPLICATION\_JSON**).

Como sólo vamos a trabajar con datos en formato Json, eliminamos para los métodos GET, POST, PUT y DELETE el formato **MediaType.APPLICATION\_XML**. A continuación guardamos el proyecto.

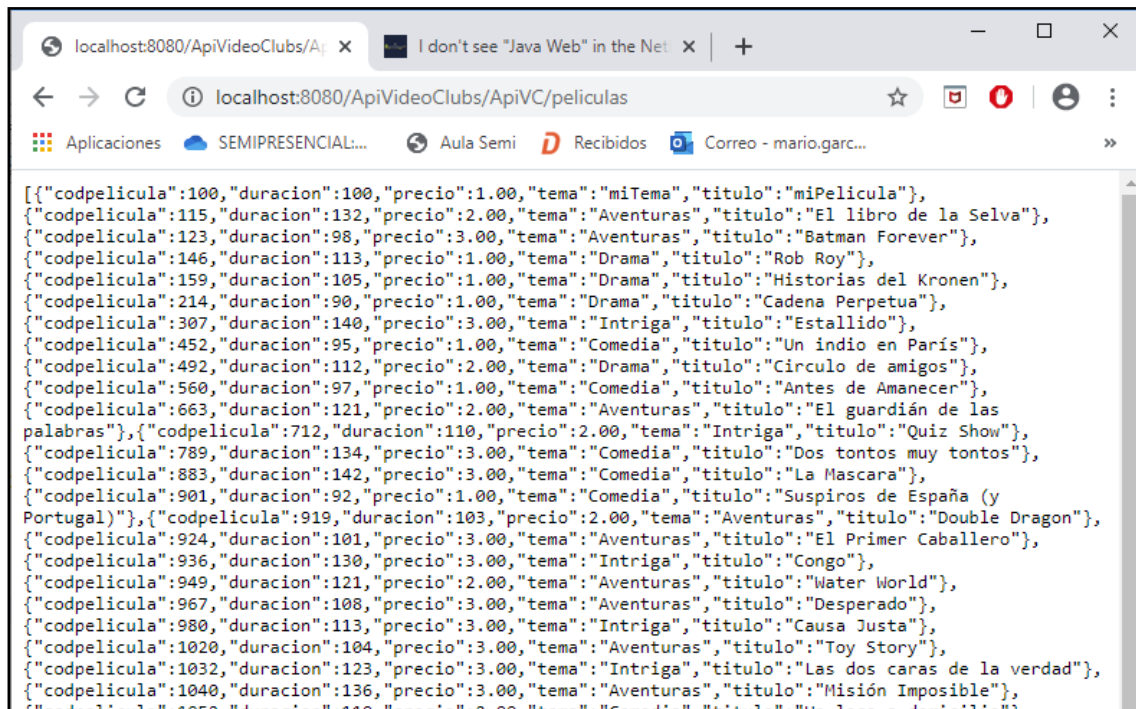
```

45  @PUT
46  @Path("{id}")
47  @Consumes({MediaType.APPLICATION_JSON})
48  public void edit(@PathParam("id") Integer id, Peliculas entity) {
49      super.edit(entity);
50  }
51
52  @DELETE
53  @Path("{id}")
54  public void remove(@PathParam("id") Integer id) {
55      super.remove(super.find(id));
56  }
57
58  @GET
59  @Path("{id}")
60  @Produces({MediaType.APPLICATION_JSON})
61  public Peliculas find(@PathParam("id") Integer id) {
62      return super.find(id);
63  }
64
65  @GET
66  @Override
67  @Produces({MediaType.APPLICATION_JSON})
68  public List<Peliculas> findAll() {
69      return super.findAll();
70  }
71
72  @GET
73  @Path("{from}/{to}")
74  @Produces({MediaType.APPLICATION_JSON})
75  public List<Peliculas> findRange(@PathParam("from") Integer from, @
76      return super.findRange(new int[]{from, to});
77

```

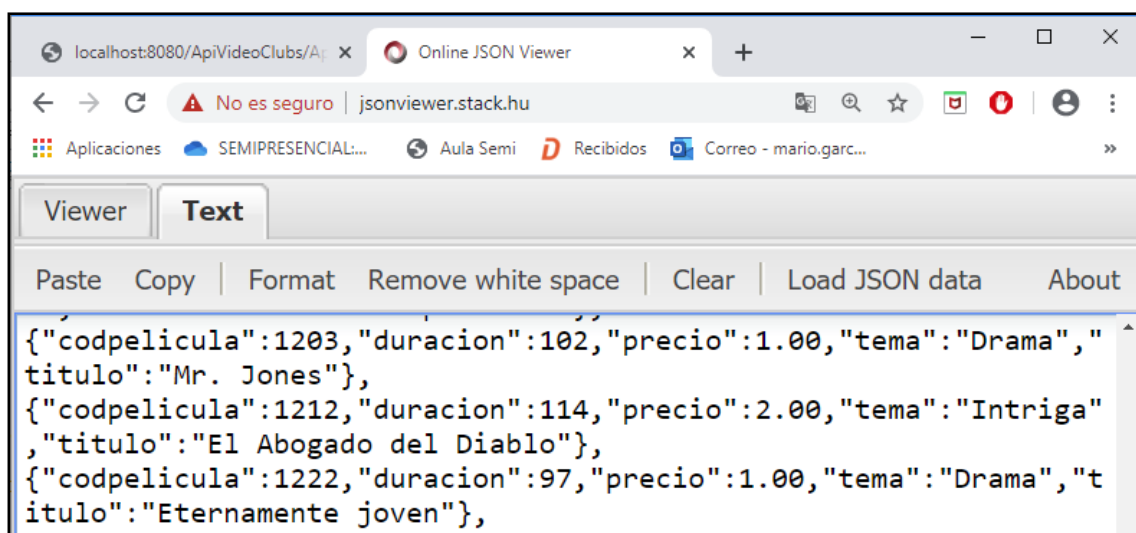


Una vez finalizado, pulsamos botón derecho sobre el proyecto y seleccionamos Deploy (Desplegar). Si queremos solicitar a la API REST el método **GET** del servicio **películas** escribiremos en el navegador la URL **http://localhost:8080/ApiVideoClubs/ApiVC/películas**. Comprobamos que muestra el listado (GET) de todas las películas la de tabla películas en formato JSON.



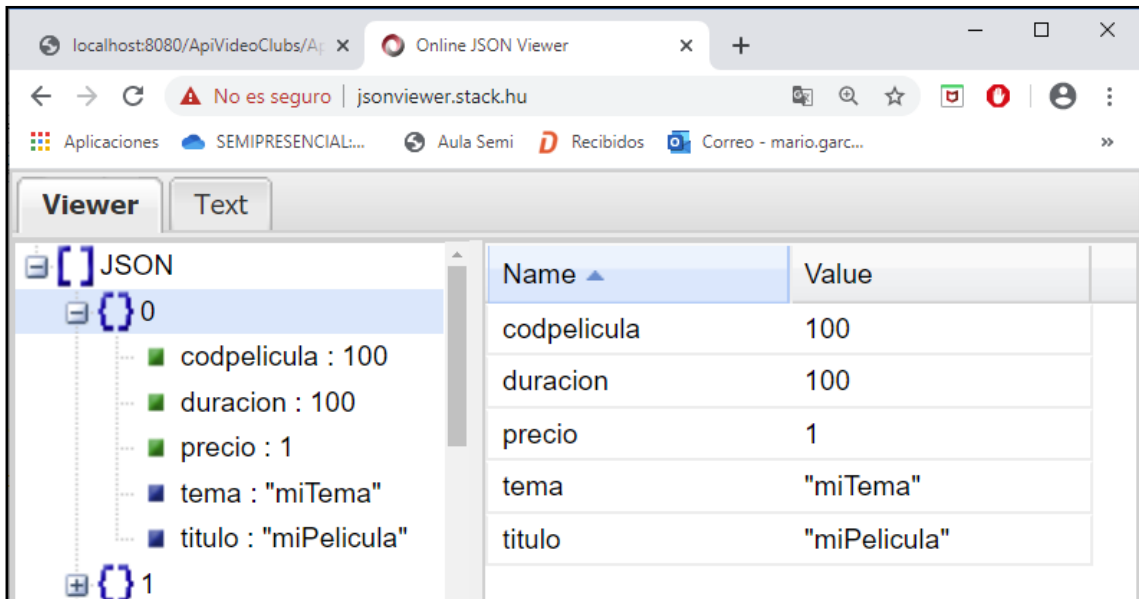
```
[{"codpelicula":100,"duracion":100,"precio":1.00,"tema":"miTema","titulo":"miPelicula"}, {"codpelicula":115,"duracion":132,"precio":2.00,"tema":"Aventuras","titulo":"El libro de la Selva"}, {"codpelicula":123,"duracion":98,"precio":3.00,"tema":"Aventuras","titulo":"Batman Forever"}, {"codpelicula":146,"duracion":113,"precio":1.00,"tema":"Drama","titulo":"Rob Roy"}, {"codpelicula":159,"duracion":105,"precio":1.00,"tema":"Drama","titulo":"Historias del Kronen"}, {"codpelicula":214,"duracion":90,"precio":1.00,"tema":"Drama","titulo":"Cadena Perpetua"}, {"codpelicula":307,"duracion":140,"precio":3.00,"tema":"Intriga","titulo":"Estallido"}, {"codpelicula":452,"duracion":95,"precio":1.00,"tema":"Comedia","titulo":"Un indio en París"}, {"codpelicula":492,"duracion":112,"precio":2.00,"tema":"Drama","titulo":"Circulo de amigos"}, {"codpelicula":560,"duracion":97,"precio":1.00,"tema":"Comedia","titulo":"Antes de Amanecer"}, {"codpelicula":663,"duracion":121,"precio":2.00,"tema":"Aventuras","titulo":"El guardián de las palabras"}, {"codpelicula":712,"duracion":110,"precio":2.00,"tema":"Intriga","titulo":"Quiz Show"}, {"codpelicula":789,"duracion":134,"precio":3.00,"tema":"Comedia","titulo":"Dos tontos muy tontos"}, {"codpelicula":883,"duracion":142,"precio":3.00,"tema":"Comedia","titulo":"La Mascara"}, {"codpelicula":901,"duracion":92,"precio":1.00,"tema":"Comedia","titulo":"Suspiros de España (y Portugal)"}, {"codpelicula":919,"duracion":103,"precio":2.00,"tema":"Aventuras","titulo":"Double Dragon"}, {"codpelicula":924,"duracion":101,"precio":3.00,"tema":"Aventuras","titulo":"El Primer Caballero"}, {"codpelicula":936,"duracion":130,"precio":3.00,"tema":"Intriga","titulo":"Congo"}, {"codpelicula":949,"duracion":121,"precio":2.00,"tema":"Aventuras","titulo":"Water World"}, {"codpelicula":967,"duracion":108,"precio":3.00,"tema":"Aventuras","titulo":"Desperado"}, {"codpelicula":980,"duracion":113,"precio":3.00,"tema":"Intriga","titulo":"Causa Justa"}, {"codpelicula":1020,"duracion":104,"precio":3.00,"tema":"Aventuras","titulo":"Toy Story"}, {"codpelicula":1032,"duracion":123,"precio":3.00,"tema":"Intriga","titulo":"Las dos caras de la verdad"}, {"codpelicula":1040,"duracion":136,"precio":3.00,"tema":"Aventuras","titulo":"Misión Imposible"}, {"codpelicula":1052,"duracion":110,"precio":2.00,"tema":"Comedia","titulo":"Un loco a domicilio"}]
```

Si queremos mostrar estos datos formateados podemos copiar el resultado en formato JSON que nos proporciona la API y pegar en la pestaña **Text** de la web <http://jsonviewer.stack.hu/>



```
{ "codpelicula":1203,"duracion":102,"precio":1.00,"tema":"Drama", "titulo":"Mr. Jones"}, {"codpelicula":1212,"duracion":114,"precio":2.00,"tema":"Intriga", "titulo":"El Abogado del Diablo"}, {"codpelicula":1222,"duracion":97,"precio":1.00,"tema":"Drama", "titulo":"Eternamente joven"},
```

Para finalizar pulsamos sobre la pestaña **Viewer** para obtener el resultado Json en formato navegable.

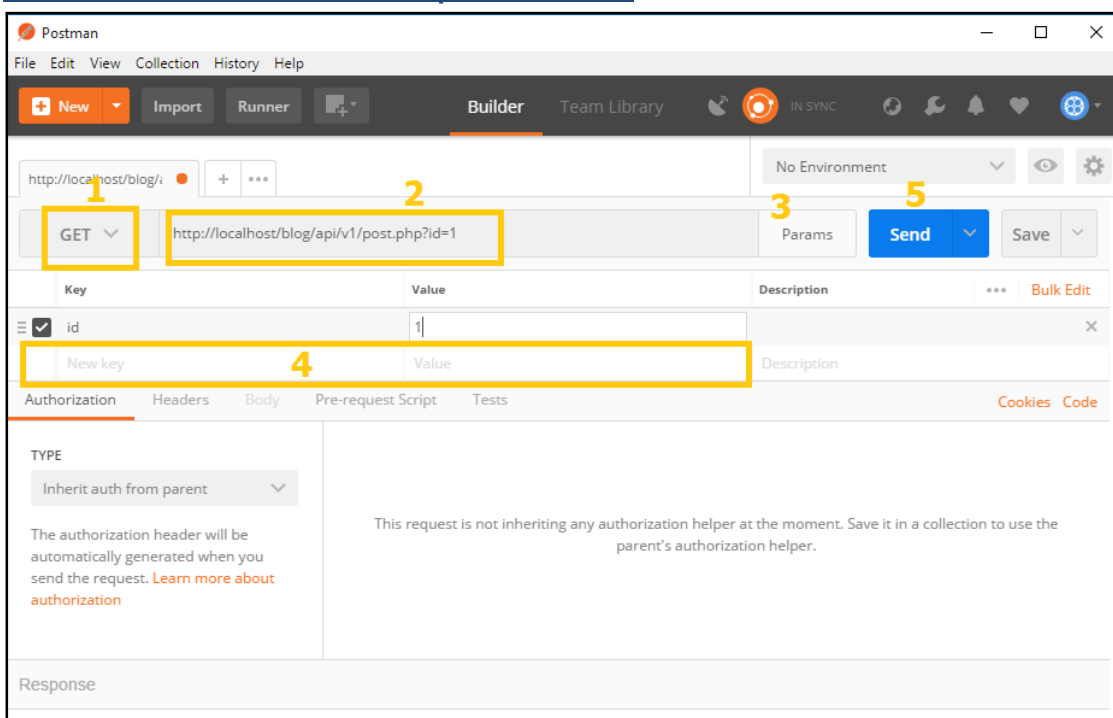


Puesto que existen métodos HTTP que no podemos probar desde el navegador, vamos a utilizar la herramienta POSTMAN para probar todos los servicios de la API y comprobar que funcionan correctamente.

### 3.- Prueba de API REST con Postman.

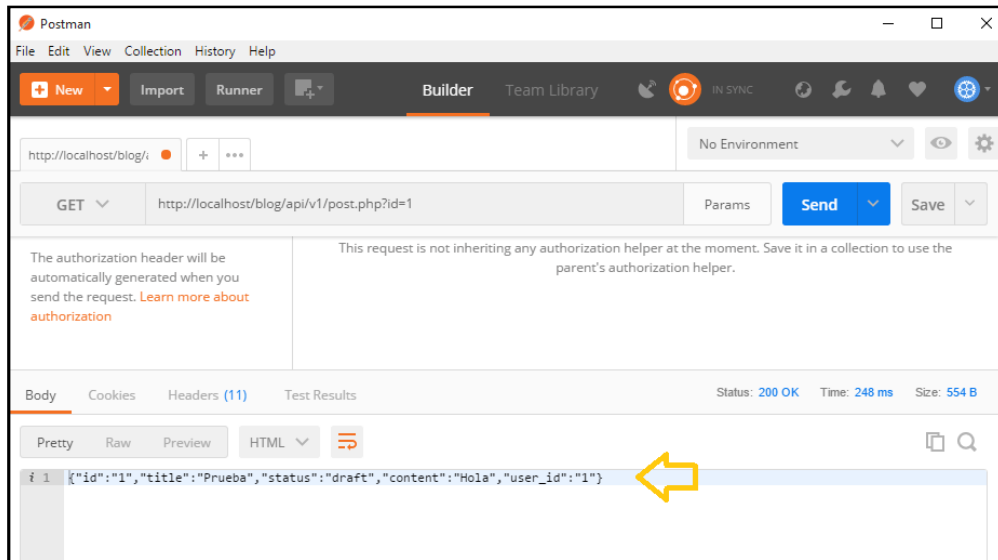
Los servicios RESTful no se pueden probar en un navegador, porque no hay forma de enviar peticiones PUT o DELETE, entonces podemos usar un programa especial como POSTMAN (<https://www.getpostman.com/downloads/>).

#### Consultar RESTful web services con parámetros GET

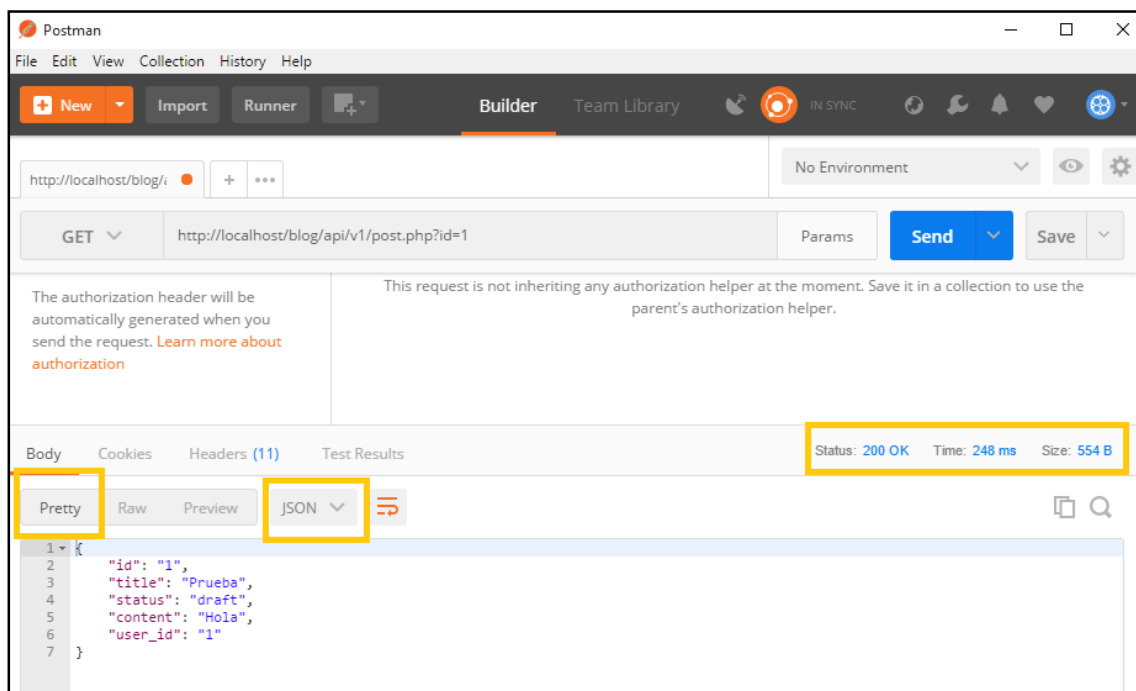


- 1.- Permite definir el método que usaremos, por ejemplo GET, POST, DELETE, etc
- 2.- Introducimos la URL del servicio RESTful
- 3.- Si presionamos el botón **Params**, se activa una sección (4) en donde puedes agregar parámetros.
- 4.- Podemos agregar parámetros que se envían junto con la dirección (URL). Debemos escribir el nombre de la clave (key) y su valor (value).
- 5.- Presionando el botón **Send** para enviar la petición.

Al ejecutar una petición a un servicio RESTful se muestra la siguiente pantalla:

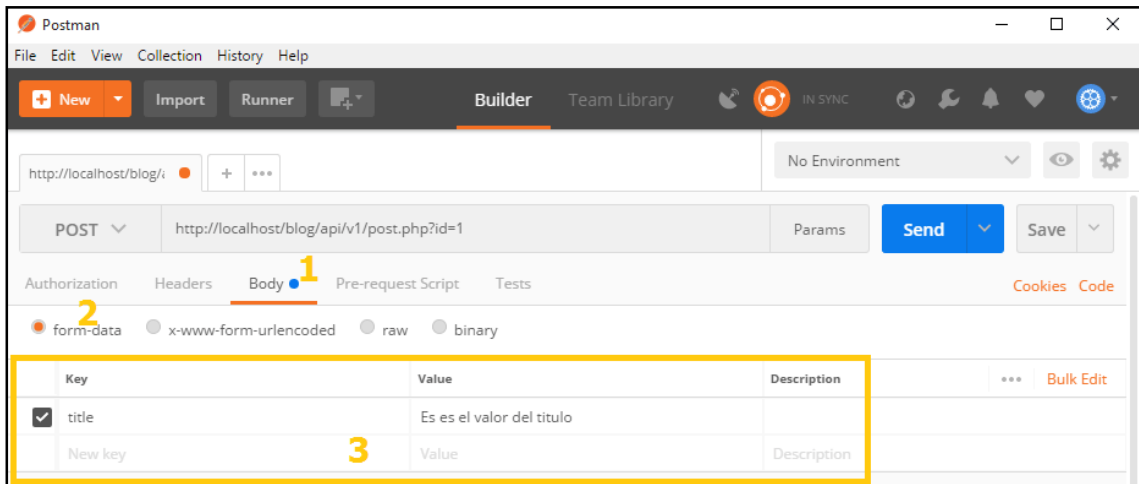


Para mostrar la respuesta en un formato más legible debemos presionar el botón **Pretty** y seleccionar el formato **JSON**. También podemos ver el código de respuesta (por ejemplo 200 ok), el tiempo que se tardó el servicio RESTful en responder y el tamaño de la respuesta



## Enviar parámetros POST (form-data) con POSTMAN

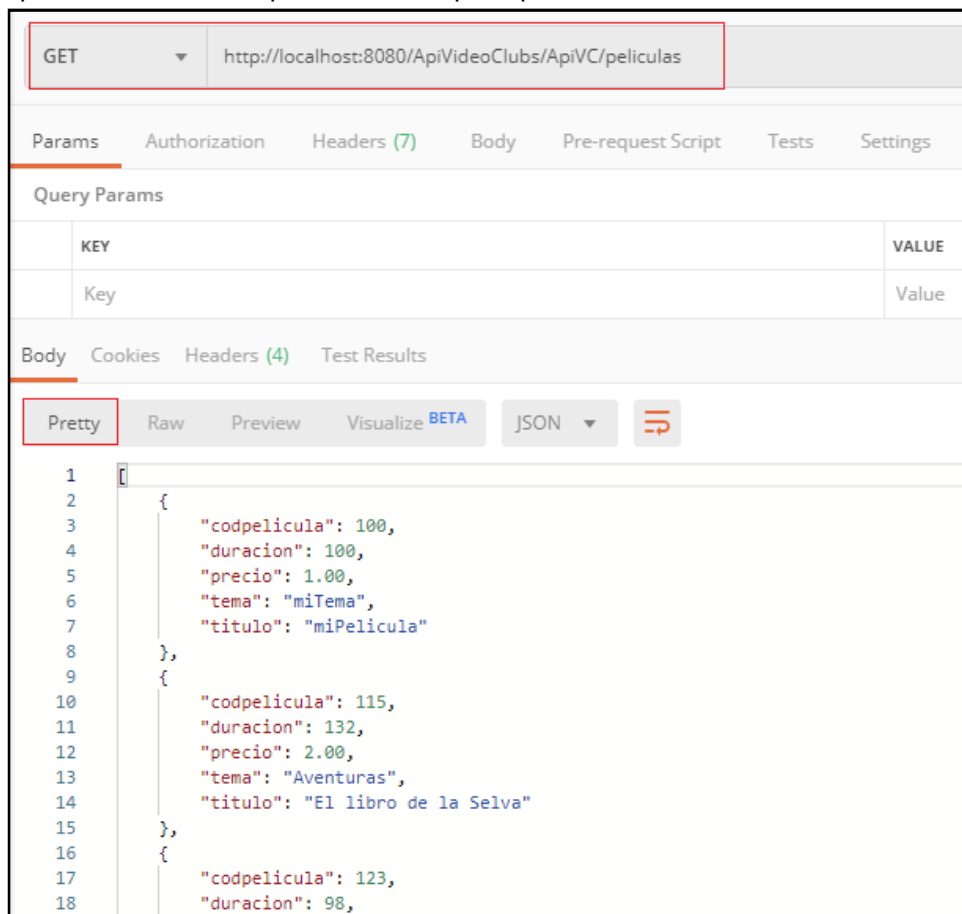
Podemos enviar parámetros por el método POST. Para ello debemos hacer clic sobre el enlace **Body**, a continuación debemos verificar que este seleccionada la opción **form-data**. Se activará la sección para introducir los parámetros. Una vez insertados los parámetros introduciendo clave y valor, pulsaremos el botón **Send**.



## Pruebas API VideoClubs.

### a) GET (Listado)

URL: <http://localhost:8080/ApiVideoClubs/ApiVC/peliculas>



## b) GET (Consulta)

**URL:** <http://localhost:8080/ApiVideoClubs/ApiVC/peliculas/115>

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** <http://localhost:8080/ApiVideoClubs/ApiVC/peliculas/115>
- Query Params:** A table with 2 columns: KEY and VALUE. The first row contains 'Key' and 'Value'.
- Body:** Pretty-printed JSON response:

```
1 {
2   "codpelicula": 115,
3   "duracion": 132,
4   "precio": 2.00,
5   "tema": "Aventuras",
6   "titulo": "El libro de la Selva"
7 }
```

## c) POST (Insertar)

**URL:** <http://localhost:8080/ApiVideoClubs/ApiVC/peliculas>

**Body:** formato raw y datos de la película que queremos dar de alta en formato json.

**Headers:** Json con codificación utf8 en **Content-Type**. (application/json;charset="utf-8")

**Resultado:** Obtenemos mensaje **Status: 204 No Content** para indicar que todo ha ido bien.

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** <http://localhost:8080/ApiVideoClubs/ApiVC/peliculas>
- Body:** raw format. The JSON body is:

```
1 {
2   "codpelicula": 8000,
3   "titulo": "Mientras dure la guerra",
4   "tema": "Drama",
5   "duracion": 150,
6   "precio": 4.25
7 }
```
- Status:** 204 No Content
- Time:** 45ms

POST ▼ http://localhost:8080/ApiVideoClubs/ApiVC/peliculas

Params Authorization **Headers (9)** Body ● Pre-request Script Tests Set

▼ Headers (1)

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	Content-Type	application/json;charset="utf-8"	
	Key	Value	Description

#### d) PUT (Actualizar)

**URL:** http://localhost:8080/ApiVideoClubs/ApiVC/peliculas/8000

**Body:** formato raw y datos de la película que queremos modificar en formato json.

**Headers:** Json con codificación utf8 en **Content-Type**. (application/json;charset="utf-8")

**Resultado:** Obtenemos mensaje **Status: 204 No Content** para indicar que todo ha ido bien.

PUT ▼ http://localhost:8080/ApiVideoClubs/ApiVC/peliculas/8000

Params Authorization Headers (1) **Body ●** Pre-request Script Tests Set

● none ● form-data ● x-www-form-urlencoded **● raw** ● binary ● GraphQL

```

1 {
2   "codpelicula": 8000,
3   "titulo": "Mientras dure la guerra",
4   "tema": "Drama",
5   "duracion": 120,
6   "precio": 2.75
7 }

```

PUT ▼ http://localhost:8080/ApiVideoClubs/ApiVC/peliculas/8000

Params Authorization **Headers (9)** Body ● Pre-request Script Tests Set

▼ Headers (1)

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	Content-Type	application/json;charset="utf-8"	
	Key	Value	Description

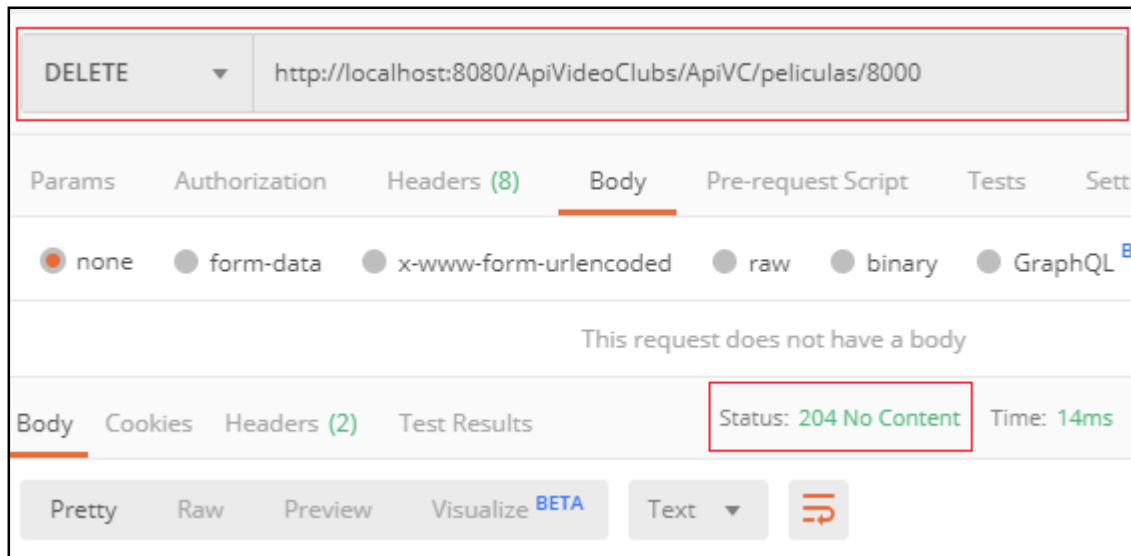
► Temporary Headers (8) ⓘ

Body Cookies Headers (2) Test Results **Status: 204 No Content** Time: 22ms

#### e) DELETE (Eliminar)

**URL:** http://localhost:8080/ApiVideoClubs/ApiVC/peliculas/8000

**Resultado:** Obtenemos mensaje **Status: 204 No Content** para indicar que todo ha ido bien.



#### 4.- Librerías y clases para aplicación cliente (HTTP).

La Clase `URLConnection` descende de la clase `URLConnection`, clase utilizada para leer información de una url específica. `URLConnection` es usado para consumir servicios web, realizando solicitudes a través del protocolo

`java.net`

### Class `URLConnection`

`java.lang.Object`

└ `java.net.URLConnection`

└ `java.net.HttpURLConnection`

HTTP. Se encuentra dentro del paquete `java.net` que es una api estándar de java. Cada instancia de `URLConnection` se utiliza para realizar una única solicitud. Dispone de los siguientes atributos y constantes:

static int	<code>HTTP_CREATED</code> HTTP Status-Code 201: Created.
static int	<code>HTTP_FORBIDDEN</code> HTTP Status-Code 403: Forbidden.
static int	<code>HTTP_INTERNAL_ERROR</code> HTTP Status-Code 500: Internal Server Error.
static int	<code>HTTP_OK</code> HTTP Status-Code 200: OK.
protected String	<code>method</code> The HTTP method (GET,POST,PUT,etc.).
protected int	<code>responseCode</code> An int representing the three digit HTTP Status-Code.
protected String	<code>responseMessage</code> The HTTP response message.

<b>Fields inherited from class java.net.URLConnection</b>
<a href="#">allowUserInteraction</a> , <a href="#">connected</a> , <a href="#">doInput</a> , <a href="#">doOutput</a> , <a href="#">ifModifiedSince</a> , <a href="#">url</a> , <a href="#">useCaches</a>

Además dispone de los siguientes métodos:

Method Summary	
abstract void	<a href="#">disconnect()</a> Indicates that other requests to the server are unlikely in the near future.
<a href="#">InputStream</a>	<a href="#">getErrorStream()</a> Returns the error stream if the connection failed but the server sent useful data nonetheless.
int	<a href="#">getResponseCode()</a> Gets the status code from an HTTP response message.
<a href="#">String</a>	<a href="#">getResponseMessage()</a> Gets the HTTP response message, if any, returned along with the response code from a server.
void	<a href="#">setRequestMethod(String method)</a> Set the method for the URL request, one of: GET POST PUT DELETE
<b>Methods inherited from class java.net.URLConnection</b>	
<a href="#">addRequestProperty</a> , <a href="#">connect</a> , <a href="#">setConnectTimeout</a> , <a href="#">setReadTimeout</a> , <a href="#">setRequestProperty</a> , <a href="#">setDoInput</a> , <a href="#">setDoOutput</a>	

Dispone de un constructor para crear instancias de la clase HttpURLConnection que necesita como parámetro una URL.

Constructor Summary	
<b>protected</b>	<a href="#">HttpURLConnection(URL u)</a> Constructor for the HttpURLConnection.

### Peticiones GET (Listado).

```
package consumoapi vc;

import java.io.*;
import java.net.*;

public class ConsumoApi VC {

    public static void main(String[] args) throws MalformedURLException, IOException {
        //Creamos Conexión
        URL url = new URL("http://localhost:8080/ApiVideoClubs/ApiVC/peliculas/");
        HttpURLConnection conexion = (HttpURLConnection) url.openConnection();
        // Establecemos Método HTTP y tipo de datos para intercambio de información
        conexion.setRequestMethod("GET");
        conexion.setRequestProperty("Accept", "application/json");

        // Obtenemos el código de respuesta.
        int codigoRespuesta = conexion.getResponseCode();
        System.out.println("Respuesta: " + codigoRespuesta);
    }
}
```



```

// Creamos el buffer de datos sobre la conexión
BufferedReader br =
    new BufferedReader(new InputStreamReader(conexion.getInputStream()));

// Variable que almacenará la salida de la petición
String salida;
// Mientras hay datos va leyendo líneas de datos
while ((salida = br.readLine()) != null) {
    System.out.println(salida);
}
// Cerramos el flujo de datos
br.close();
// Cerramos la conexión
conexion.disconnect();
}
}

```

### Salida:

```

Respuesta: 200
[{"codpeli":100,"duracion":100,"precio":1.00,"tema":"mi Tema","titulo":"mi Pelicula"}, {"codpeli":123,"duracion":
98,"precio":3.00,"tema":"Aventuras","titulo":"Batman
Forever"}, {"codpeli":146,"duracion":113,"precio":1.00,"tema":"Drama","titulo":"Rob
Roy"}, {"codpeli":159,"duracion":105,"precio":1.00,"tema":"Drama","titulo":"Historias del
Kronen"}, {"codpeli":214,"duracion":90,"precio":1.00,"tema":"Drama","titulo":"Cadena
Perpetua"}, {"codpeli":307,"duracion":140,"precio":3.00,"tema":"Intriga","titulo":"Estalido"}, {"codpeli":452,"
duracion":95,"precio":1.00,"tema":"Comedia","titulo":"Un indio en
París"}, {"codpeli":492,"duracion":112,"precio":2.00,"tema":"Drama","titulo":"Círculo de
amigos"}, {"codpeli":560,"duracion":97,"precio":1.00,"tema":"Comedia","titulo":"Antes de
Amanecer"}, {"codpeli":663,"duracion":121,"precio":2.00,"tema":"Aventuras","titulo":"El guardián de las
palabras"}, {"codpeli":712,"duracion":110,"precio":2.00,"tema":"Intriga","titulo":"Quiz
Show"}, {"codpeli":789,"duracion":134,"precio":3.00,"tema":"Comedia","titulo":"Dos tontos muy
tontos"}, {"codpeli":883,"duracion":142,"precio":3.00,"tema":"Comedia","titulo":"La
Mascara"}, {"codpeli":901,"duracion":92,"precio":1.00,"tema":"Comedia","titulo":"Suspiros de España (y
Portugal)"}, {"codpeli":919,"duracion":103,"precio":2.00,"tema":"Aventuras","titulo":"Double
Dragon"}, {"codpeli":924,"duracion":101,"precio":3.00,"tema":"Aventuras","titulo":"El Primer
Caballero"}, {"codpeli":936,"duracion":130,"precio":3.00,"tema":"Intriga","titulo":"Congo"}, {"codpeli":949,"dur
acion":121,"precio":2.00,"tema":"Aventuras","titulo":"Water
World"}, {"codpeli":967,"duracion":108,"precio":3.00,"tema":"Aventuras","titulo":"Desperado"}, {"codpeli":980,"d
uracion":113,"precio":3.00,"tema":"Intriga","titulo":"Causa
Justa"}, {"codpeli":1020,"duracion":104,"precio":3.00,"tema":"Aventuras","titulo":"Toy
Story"}, {"codpeli":1032,"duracion":123,"precio":3.00,"tema":"Intriga","titulo":"Las dos caras de la
verdad"}, {"codpeli":1040,"duracion":136,"precio":3.00,"tema":"Aventuras","titulo":"Misión
Imposible"}, {"codpeli":1052,"duracion":110,"precio":2.00,"tema":"Comedia","titulo":"Un loco a
domicilio"}, {"codpeli":1065,"duracion":95,"precio":2.00,"tema":"Comedia","titulo":"Como conquistar
Hollywood"}, {"codpeli":1072,"duracion":112,"precio":3.00,"tema":"Drama","titulo":"Tres
Deseos"}, {"codpeli":1083,"duracion":127,"precio":1.00,"tema":"Drama","titulo":"Cautivos"}, {"codpeli":1090,"dur
acion":133,"precio":3.00,"tema":"Drama","titulo":"Días de
Fortuna"}, {"codpeli":1102,"duracion":111,"precio":1.00,"tema":"Comedia","titulo":"El Presidente y Miss
Wade"}, {"codpeli":1109,"duracion":101,"precio":1.00,"tema":"Comedia","titulo":"Sabrina y sus
amores"}, {"codpeli":1117,"duracion":124,"precio":2.00,"tema":"Drama","titulo":"Una Proposición
Indecente"}, {"codpeli":1123,"duracion":105,"precio":3.00,"tema":"Comedia","titulo":"Una Jaula de
Grillos"}, {"codpeli":1134,"duracion":119,"precio":2.00,"tema":"Drama","titulo":"La letra
Escarlata"}, {"codpeli":1140,"duracion":98,"precio":3.00,"tema":"Drama","titulo":"City
Hall"}, {"codpeli":1153,"duracion":103,"precio":2.00,"tema":"Comedia","titulo":"Two
Much"}, {"codpeli":1162,"duracion":116,"precio":3.00,"tema":"Comedia","titulo":"Aventura, Operación
África"}, {"codpeli":1170,"duracion":100,"precio":1.00,"tema":"Drama","titulo":"Todo por un
Sueño"}, {"codpeli":1188,"duracion":131,"precio":2.00,"tema":"Aventuras","titulo":"Tierra
peligrosa"}, {"codpeli":1193,"duracion":114,"precio":1.00,"tema":"Aventuras","titulo":"El informe
pelicano"}, {"codpeli":1203,"duracion":102,"precio":1.00,"tema":"Drama","titulo":"Mr.
Jones"}, {"codpeli":1212,"duracion":114,"precio":2.00,"tema":"Intriga","titulo":"El Abogado del
Diablo"}, {"codpeli":1222,"duracion":97,"precio":1.00,"tema":"Drama","titulo":"Eternamente
joven"}, {"codpeli":1234,"duracion":115,"precio":3.00,"tema":"Drama","titulo":"Ojo por
ojo"}, {"codpeli":1241,"duracion":141,"precio":3.00,"tema":"Drama","titulo":"Leyendas de
Pasión"}, {"codpeli":1253,"duracion":103,"precio":2.00,"tema":"Comedia","titulo":"Esperando un
Respiro"}, {"codpeli":1267,"duracion":129,"precio":3.00,"tema":"Intriga","titulo":"La
red"}, {"codpeli":1273,"duracion":133,"precio":3.00,"tema":"Aventuras","titulo":"Eraser"}, {"codpeli":1288,"dura
cion":100,"precio":1.00,"tema":"Aventuras","titulo":"Salto al
vacío"}, {"codpeli":1295,"duracion":119,"precio":2.00,"tema":"Aventuras","titulo":"Golden Eye"}]
BUILD SUCCESSFUL (total time: 0 seconds)

```

### Peticiones GET (Consulta).

```
package consumoapi vc;

import java.io.*;
import java.net.*;

public class ConsumoApi VC {

    public static void main(String[] args) throws Mal formedURLException, IOException
    {
        //Creamos Conexión
        URL url = new URL("http://localhost:8080/ApiVi deoCl ubs/Api VC/pel i cul as/100");
        HttpURLConnection conexion = (HttpURLConnection) url.openConnection();
        // Establecemos Método HTTP y tipo de datos para intercambio de información
        conexion.setRequestMethod("GET");
        conexion.setRequestProperty("Accept", "appl i cati on/j son");

        // Obtenemos el código de respuesta.
        int codigoRespuesta = conexion.getResponseCode();
        System.out.println("Respuesta: " + codi goRespuesta);

        // Creamos el buffer de datos sobre la conexión
        BufferedReader br =
            new BufferedReader(new InputStreamReader(conexion.getInputStream()));

        // Variable que almacenará la salida de la petición
        String salida;
        // Mientras hay datos va leyendo líneas de datos
        while ((salida = br.readLine()) != null) {
            System.out.println(salida);
        }
        // Cerramos el flujo de datos
        br.close();
        // Cerramos la conexión
        conexion.disconnect();
    }
}
```

Salida:

```
run:
Respuesta: 200
{"codpel i cul a": 100, "duraci on": 100, "preci o": 1.00, "tema": "mi Tema", "ti tul o": "mi Pel i cul a"
}
BUILD SUCCESSFUL (total time: 0 seconds)
```

### Peticiones DELETE (Eliminar).

```
package consumoapi vc;

import java.io.*;
import java.net.*;

public class ConsumoApi VC {

    public static void main(String[] args) throws Mal formedURLException, IOException
    {
```

```

//Creamos Conexión
URL url = new URL("http://localhost:8080/APIVCS/api/películas/10");
URLConnection conexion = (URLConnection) url.openConnection();
// Establecemos Método HTTP y tipo de datos para intercambio de información
conexion.setRequestMethod("DELETE");
conexion.setRequestProperty("Accept", "application/json");
// Obtenemos el código de respuesta.
int codigoRespuesta = conexion.getResponseCode();
System.out.println("Respuesta: " + codigoRespuesta);
// Cerramos la conexión
conexion.disconnect();
}

```

Salida:

```

run:
Respuesta: 204
BUILD SUCCESSFUL (total time: 0 seconds)

```

### Peticiones POST (Insertar).

```

package consumoapivc;
import java.io.*;
import java.net.*;

public class ConsumoApiVC {
    public static void main(String[] args) throws MalformedURLException, IOException {
        //Creamos Conexión
        URL url = new URL("http://localhost:8080/APIVCS/api/películas/");
        URLConnection conexion = (URLConnection) url.openConnection();
        // Establecemos que se va a enviar información, el método HTTP
        // y tipo de datos para intercambio de información
        conexion.setDoOutput(true);
        conexion.setRequestMethod("POST");
        conexion.setRequestProperty("Content-Type", "application/json");

        // Creamos el json
        String json = "{\"codpelícula\":10, \"título\":\"Mientras dure la guerra\", \"tema\":\"Drama\", \"duración\":150, \"precio\":4.50}";

        // Escribimos información json que enviaremos a la petición
        // sobre el flujo de datos de salida de la conexión
        OutputStream os = conexion.getOutputStream();
        os.write(json.getBytes());
        os.flush();

        // Obtenemos el código de respuesta.
        int codigoRespuesta = conexion.getResponseCode();
        System.out.println("Respuesta: " + codigoRespuesta);
        // Cerramos la conexión
        conexion.disconnect();
    }
}

```

Salida:

```
run:
Respuesta: 204
BUILD SUCCESSFUL (total time: 0 seconds)
```

### Peticiones PUT (Actualizar).

```
package consumoapi.vc;
import java.io.*;
import java.net.*;

public class ConsumoApiVC {
    public static void main(String[] args) throws MalformedURLException, IOException
    {
        //Creamos Conexión
        URL url = new URL("http://localhost:8080/APIVCS/api/películas/10");
        HttpURLConnection conexion = (HttpURLConnection) url.openConnection();
        // Establecemos que se va a enviar información, el método HTTP
        // y tipo de datos para intercambio de información
        conexion.setDoOutput(true);
        conexion.setRequestMethod("PUT");
        conexion.setRequestProperty("Content-Type", "application/json");

        // Creamos el json
        String json = "{\"codpelícula\":10, \"título\":\"Mientras dure la guerra\", \"
            + \"tema\":\"Drama\", \"duración\":120, \"precio\":4.25}";

        // Escribimos información json que enviaremos a la petición
        // sobre el flujo de datos de salida de la conexión
        OutputStream os = conexion.getOutputStream();
        os.write(json.getBytes());
        os.flush();

        // Obtenemos el código de respuesta.
        int codigoRespuesta = conexion.getResponseCode();
        System.out.println("Respuesta: " + codigoRespuesta);

        // Cerramos la conexión
        conexion.disconnect();
    }
}
```

Salida

```
run:
Respuesta: 204
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Ejemplo:** Consumo de la API OpenWeatherMap que proporciona en formato json información del tiempo de una localización. Precisa registro para que se nos proporcione la API Key para su consumo. Además precisa la librería org.json.jar para poder convertir el json recibido desde la url en un objeto que podamos tratar para mostrar la información deseada.

```

package consumoapi;
import java.io.*;
import java.net.*;
import org.json.JSONObject;

class ConsumoAPI {
    public static void main(String[] args) throws Exception {
        String url =
"http://api.openweathermap.org/data/2.5/weather?q=valencia&units=metric&appid
=a77a241542b9ed8ac5b8e533f10c2f0a";
        String respuesta = "";
        respuesta = petitionHttpGet(url);
        //System.out.println("La respuesta es:\n" + respuesta);
        JSONObject json = new JSONObject(respuesta);
        String lugar = json.getString("name");
        double tmin = json.getJSONObject("main").getDouble("temp_min");
        double tmax = json.getJSONObject("main").getDouble("temp_max");
        double temp = json.getJSONObject("main").getDouble("temp");
        double hum = json.getJSONObject("main").getDouble("humidity");
        double pres = json.getJSONObject("main").getDouble("pressure");
        System.out.println("Lugar: " + lugar);
        System.out.println("Temp: " + temp);
        System.out.println("Temp. Min: " + tmin);
        System.out.println("Temp. Max: " + tmax);
        System.out.println("Humedad: " + hum);
        System.out.println("Presión: " + pres);
    }
    public static String petitionHttpGet(String urlParaVisitar)
        throws MalformedURLException, IOException {
        // Esto es lo que vamos a devolver
        String resultado = "";
        // Crear un objeto de tipo URL
        URL url = new URL(urlParaVisitar);
        // Abrir la conexión e indicar que será de tipo GET
        HttpURLConnection conexion =
            (HttpURLConnection) url.openConnection();
        conexion.setRequestMethod("GET");
        // Búferes para leer
        BufferedReader rd =
            new BufferedReader(new InputStreamReader(conexion.getInputStream()));
        String linea;
        // Mientras BufferedReader pueda leer, agregar contenido a resultado
        while ((linea = rd.readLine()) != null) {
            resultado += linea;
        }
        // Cerrar el BufferedReader
        rd.close();
        // Regresar resultado, pero como cadena, no como StringBuilder
        return resultado;
    }
}

```

```
{ "coord": { "lon": -0.38, "lat": 39.47 }, "weather": [ { "id": 801, "main": "Clouds", "description": "few clouds", "icon": "02n" } ], "base": "stations", "main": { "temp": 11.48, "feels_like": 8.83, "temp_min": 10, "temp_max": 12.22, "pressure": 1032, "humidity": 71 }, "visibility": 10000, "wind": { "speed": 2.6, "deg": 150 }, "clouds": { "all": 20 }, "dt": 1577726865, "sys": { "type": 1, "id": 6432, "country": "ES", "sunrise": 1577690480, "sunset": 1577724364 }, "timezone": 3600, "id": 2509954, "name": "Valencia", "cod": 200 }
```

```

{
  "coord": {
    "lon": 0.3763,
    "lat": 39.4699
  },
  "weather": [
    {
      "id": 800,
      "main": "clear",
      "description": "clear sky",
      "icon": "01d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 11.48,
    "feels_like": 8.83,
    "temp_min": 10,
    "temp_max": 12.22,
    "pressure": 1032,
    "humidity": 71,
    "visibility": 10000
  },
  "wind": {
    "speed": 3.6,
    "deg": 140
  },
  "clouds": {
    "all": 0
  },
  "dt": 1577726865,
  "sys": {
    "timezone": 3600,
    "id": 2509954,
    "name": "Valencia",
    "cod": 200
  }
}

```

```
JSONObject json = new JSONObject(respuesta);
String lugar = json.getString("name");
double tmin = json.getJSONObject("main").getDouble("temp_min");
:
:
:
:
:
```

Ejecución:

```
run:  
Lugar: Valencia  
Temp: 9.61  
Temp. Min: 4.44  
Temp. Max: 12.22  
Humedad: 71.0  
Presión: 1032.0  
BUILD SUCCESSFUL (total time: 0 seconds)
```