

Lab No.2 Fundamentos De Sistemas

JSON



Por:

Joaquin David Hernandez Cardenas. 1038481420

Pablo Andres Diaz Gomez. 1214717460

Luis Alejandro Zambrano Sanchez. 1241714749

Estefania Morales Araque. 1152438479

Carlos Enrique Agudelo Giraldo. 1038410721

Profesor:

Diego Botia

Universidad de Antioquia

Medellín (ANT)

2014- 2

1. Introducción:

En este laboratorio se tratarán temas de gran importancia siguiendo un enfoque con respecto al diseño y desarrollo de páginas web y continuando con la secuencia lógica del laboratorio anteriormente realizado. Vamos a introducir el concepto de JSON como un formato de datos que nos permite compartir información entre sistemas que pueden estar implementados en lenguajes diferentes, además se harán los respectivos ejemplos demostrando las propiedades y ventajas de trabajar con esta herramienta. Se implementará la integración de Jersey y Jackson en JSON para un mejor manejo del formato JSON, además de ejemplificar estas herramientas en un ambiente de servicios web RESTful.

2. Objetivos

Generales: generar los conocimientos y habilidades necesarias con respecto al manejo y funcionalidad del formato JSON, desarrollando así en el lector y en las personas que desarrollaron el laboratorio las competencias necesarias para enfrentarse a problemas que impliquen la comunicación de información entre diferentes sistemas informáticos, algo que será muy común en su vida laboral.

Específicos:

- Manejo del IDE Netbeans para el desarrollo de proyectos con JSON.
- Generar el conocimiento suficiente con respecto a JSON.
- Mostrar el manejo de la herramienta además de adquirir las habilidades necesarias con respecto al uso e implementación de esta.
- Dar a conocer las herramientas Jersey y Jackson.
- Enseñar las virtudes de integrar Jersey y Jackson a JSON para el desarrollo para la conversión de los objetos en java
- Entender el concepto de RESTful y sus aplicaciones.
- Ejemplificar la utilización de RESTful y las ventajas de trabajar estos servicios web.

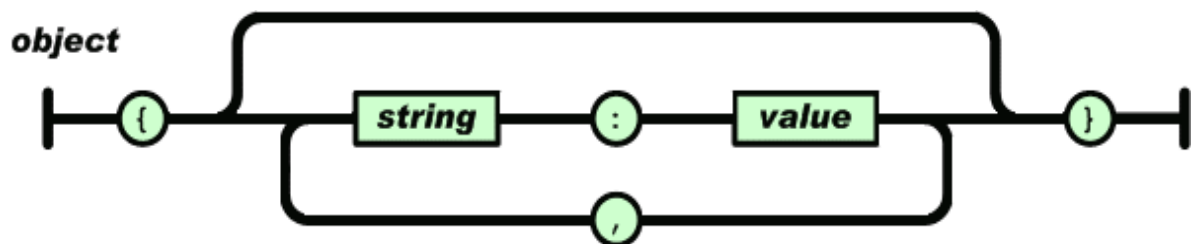
3. Marco Teórico.

En la nuestra etapa laboral encontraremos un sinfín de herramientas que nos permitirá dar las soluciones más adecuadas dependiendo de los requerimientos del problema, por tal motivo estar a la vanguardia tanto de la tecnología como de la información se hace cada vez más indispensable. El desarrollo sistemas informáticos web no es la excepción ya que para llevar a cabo esta tarea contamos con múltiples plataformas además de lenguajes que dependiendo del producto final pueden ser utilizados, permitiendo a su vez que los trabajos realizados sean mejores en términos de eficiencia y funcionamiento y más aplicados en función de lo que el cliente final desea. Competir en el mercado de desarrollo de software no es nada fácil si nos centramos en el hecho de que mundialmente miles de personas se destinan a esta tarea, pero con los conocimientos suficientes y la práctica necesaria podremos estar a la vanguardia y al nivel de muchos desarrolladores que hacen de este trabajo un arte. Por tal motivo el documento tendrá como objetivo mostrar tanto conceptual como práctico el funcionamiento de una de un formato que nos permitirá gestionar la información que se transfiere entre sistemas web de manera más sencilla y eficiente (JSON), permitiendo a su vez enfocarnos en trabajos mucho más técnicos de la misma, también se hablará acerca de las herramientas complementarias que nos permitirán una mayor eficiencia de nuestros trabajos, sin dejar a un lado los servicios web que se usarán mediante la práctica para hacer demostrativos los ejercicios.

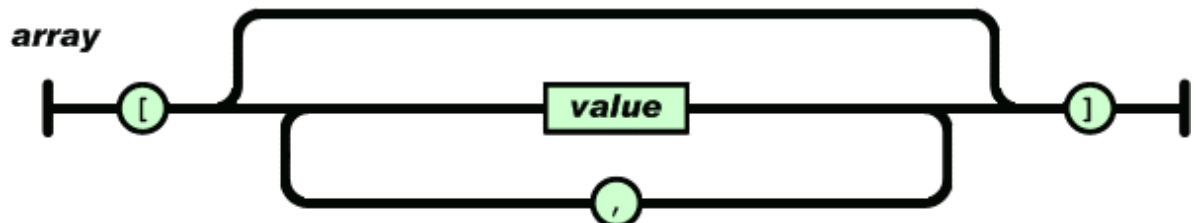
Como lo hemos mencionado anteriormente JSON (JavaScript Object Notation) es un formato de intercambio de datos, dado que la sintaxis del formato puede acoplar a casi todos los lenguajes de programación, permitiendo así la transmisión de información entre cualquier tipo de tecnología siendo esta la sintaxis ideal cuanto a transmisión de información se refiere. JSON nace como una alternativa de XML, ya que al ser un formato más liviano por la estructura que utiliza el trabajo con esta herramienta se hace más eficiente, además JSON es compatible con casi todos los lenguajes de programación que tenemos actualmente en el mercado.

JSON está apoyado en dos tipos de estructuras de datos, estas estructuras son universales para todos los lenguajes de programación, lo que quiere decir que la mayoría de lenguajes se basan de la misma forma. Una de las

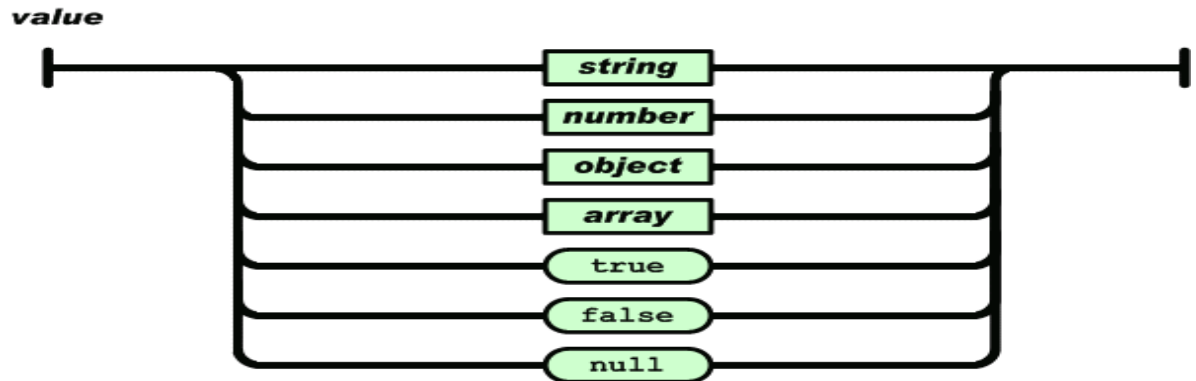
estructuras bajo la cual está basado JSON es la de colección de pares nombre/valor en una Hash Table donde el parés hace referencia a lo que se conoce comúnmente como registro de la estructura donde el nombre hace referencia a la clave y el valor hará referencia al objeto almacenado. La otra estructura sobre la cual se enfoca esta herramienta es la de lista ordenada de valores conocida más comúnmente como vector o matriz de registros. Los objetos en este formato son pares nombre/valor, por sintaxis siempre empiezan con llave izquierda y terminan con llave derecha, cada nombre estará seguido por dos puntos y su correspondiente valor y cada uno de estos está separado por una coma.



Con respecto a la matriz o vector como se dijo anteriormente esta es una colección ordenada de valores, la sintaxis para declarar una matriz sería empezando con corchete y terminando de la misma forma, los valores que ésta almacenará serán separados por comas.



Por último con respecto a los valores estos pueden ser una cadena de caracteres entre comillas dobles, también pueden ser un número, también puede ser un dato booleano, o puede estar en nulo, el trabajo bajo este concepto o estructura tiene la cualidad de que puede ser anidado.

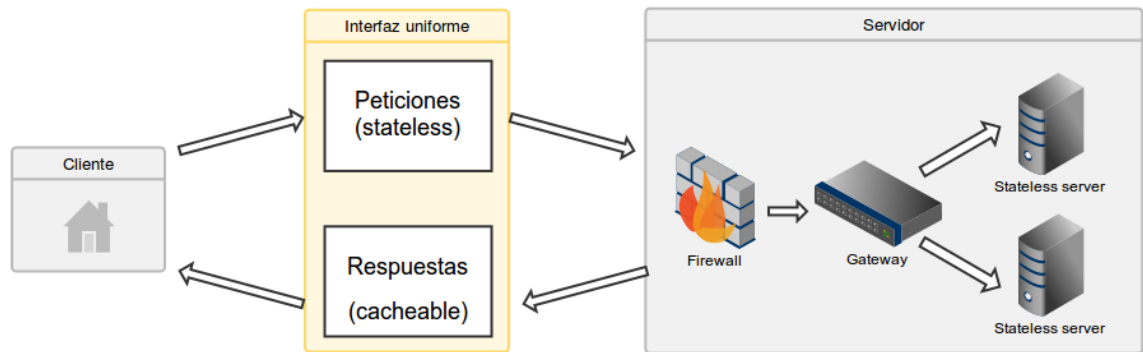


Jersey y Jackson son dos librerías de java que integradas permiten convertir objetos del formato JSON a objetos de las clases de java , e igualmente se puede hacer el proceso inversos, en el caso de Jackson que fue un api diseñada con el fin del procesamiento de JSON, esta es una de las mejores combinaciones posibles en términos de agilidad, optimización y ergonomía para los desarrolladores. Mientras de que Jersey es una API enfocada en la implementación de calidad de producción, (una API de java para los servicios web de RESTful), esta a su vez fue diseñada para permitir a los desarrolladores crear servicios web en RESTful con java. Pero cuando estas dos trabajan integradas a JSON las cosas cambian para lograr un mayor desempeño tanto del proyecto final como de cada una de estas herramientas. El funcionamiento de estas APIs aplicadas a un proyecto es el siguiente, Jersey usando Jackson es el encargado de transmitir los objetos del lenguaje de programación sobre el que se trabaja a JSON integrando así en un proyecto la parte de transmisión de objetos a una parte técnicamente menos compleja y más liviana con respecto a otros tipos de técnicas usadas para la solución de estos problemas.

Continuando con respecto al enfoque del documento, algo supremamente importante en el desarrollo de aplicaciones web y que en los últimos años ha tomado gran fuerza e ímpetus en el desarrollo de estos proyectos son los servicios web RESTful, los servicios REST (REpresentational State Transfer) son principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema incluyendo el cómo se accede a los estados de estos recursos y la forma en cómo se transfiere mediante los HTTP hacia los clientes escritos en varios lenguajes.

La implementación correcta de los RESTful sigue algunas reglas de arquitectura ya definida estas son: arquitectura cliente servidor consiste en una separación clara y concisa de los dos agentes que intervienen en el intercambio de la información el cliente y el servidor, Stateless que indica que el servidor no tiene porqué guardar los datos del cliente, cacheable

esta norma define que el servidor que sirve las peticiones del cliente debe definir algún modo de cachear dichas peticiones, para aumentar el rendimiento y escalabilidad, sistema por capas implica que el sistema no debe forzar al cliente a saber por qué capas se tramita la información y interfaz uniforme esta regla simplifica el protocolo y aumenta la escalabilidad y rendimiento del sistema ya que no dependeremos del cliente ni del servidor solo de quien haga el uso de la interfaz.



Hasta acá se presenta una de las formas posibles de transferir información entre sistemas, como se ha expresado a lo largo de este documento, es decisión del desarrollador escoger la opción más adecuada dependiendo los requerimientos y funcionalidades del trabajo a desarrollar, herramientas para este tipo de trabajos aparte de las ya estudiadas y trabajadas existen muchas, es decisión personal elegir la más adecuada de acuerdo al problema al que se le desea dar solución

4. Procedimiento

1. Conversor de XML a JSON

En este ejercicio se realiza una aplicación JAVA que permite convertir un documento en formato XML a JSON. Para esto se emplean las clases de json para java, suministradas por el profesor.

Primero se tendrá un documento XML proporcionado por el profesor, el cual lo convertiremos a Json creando una aplicación en java llamada HolaJson, colocamos las respectivas clases en el packpageorg.json

Después de esto nos retorna un objeto JSONObject que contiene los datos que provienen del archivo xml, se procede a llamar el método toJSONObject() y se compila el proyecto dando como resultado :

```
7  *
8  * @author Joaquin David
9  */
10 import org.json.JSONException;
11 import org.json.JSONObject;
12 import org.json.XML;
13
14 public class HolaM {
15     public static int PRETTY_PRINT_INDENT_FACTOR = 4;
16     public static String TEST_XML_STRING =
17         "<breakfast_menu>\n"+
18         "    <food>\n"+
19         "        <name>Belgian Waffles</name>\n"+
20         "        <price>$5.95</price>\n"+
21         "        <description>\n"+
22         "            Two of our famous Belgian Waffles with plenty of real maple syrup\n"+
23         "        </description>\n"+
24         "        <calories>650</calories>\n"+
25         "    </food>\n"+
26         "    <food>\n"+
27         "        <name>Strawberry Belgian Waffles</name>\n"+
```

main - Navigator x

Members

<empty>

HolaM

- main(String[] args)
- PRETTY_PRINT_INDENT_FACTOR: int
- TEST_XML_STRING: String

Output - HolaJson (run) x

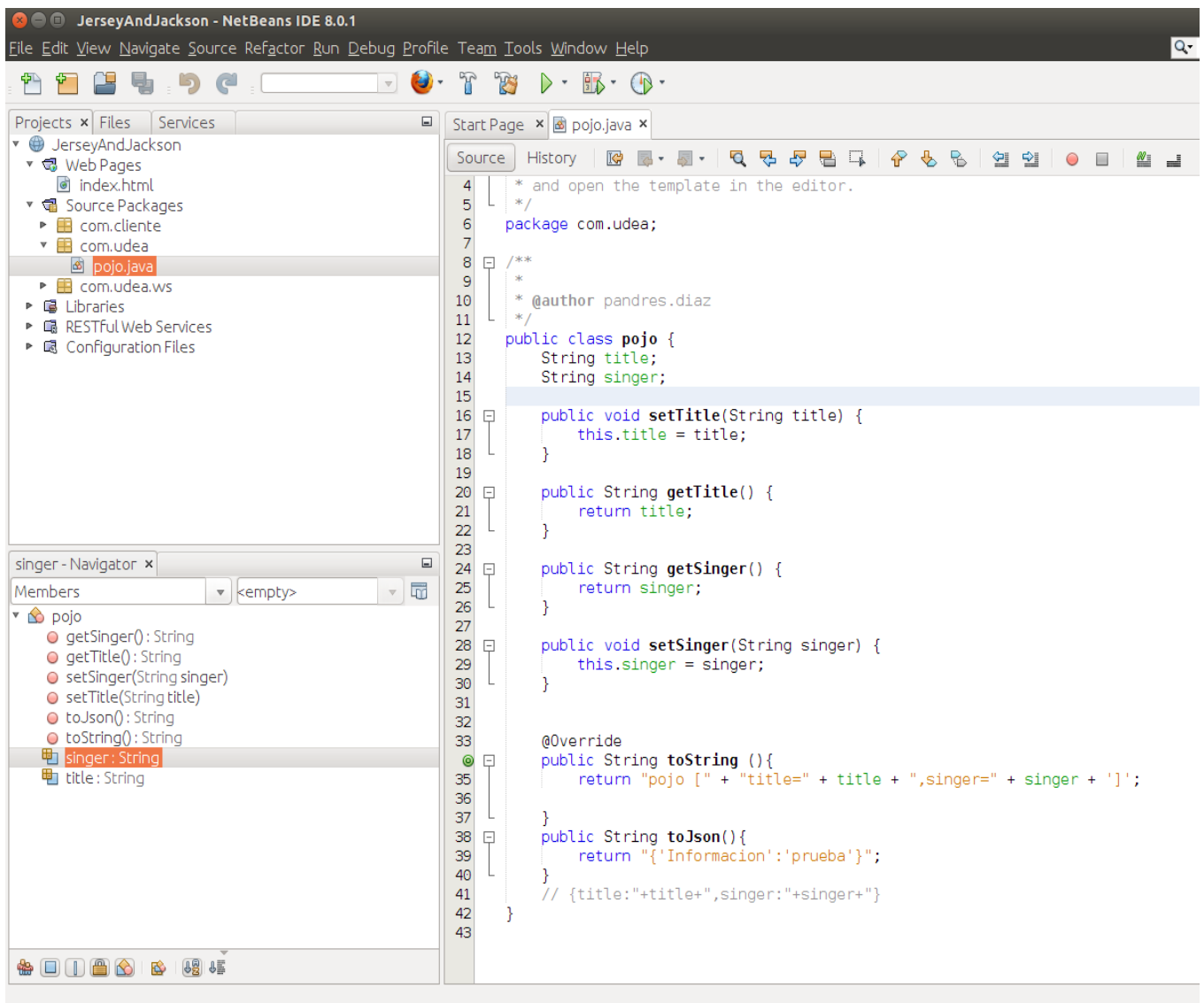
```
run:
{"breakfast_menu": {"food": [
  {
    "price": "$5.95",
    "description": "Two of our famous Belgian Waffles with plenty of real maple syrup",
    "name": "Belgian Waffles",
    "calories": 650
  },
  {
    "price": "$7.95",
    "description": "Light Belgian waffles covered with strawberries and whipped cream",
    "name": "Strawberry Belgian Waffles",
    "calories": 900
  }
]}
BUILD SUCCESSFUL (total time: 0 seconds)
```


2. Ejemplo de JSON con Jersey + Jackson

El API Jersey utiliza Jackson para convertir objetos desde y hacia JSON. En este ejemplo se convertirá un objeto llamado “track” en formato JSON y será retornado al usuario.

Primero que todo procedemos a crear una aplicación web en Netbeans, utilizando la plataforma JEE7 y por medio del servidor Glassfish.

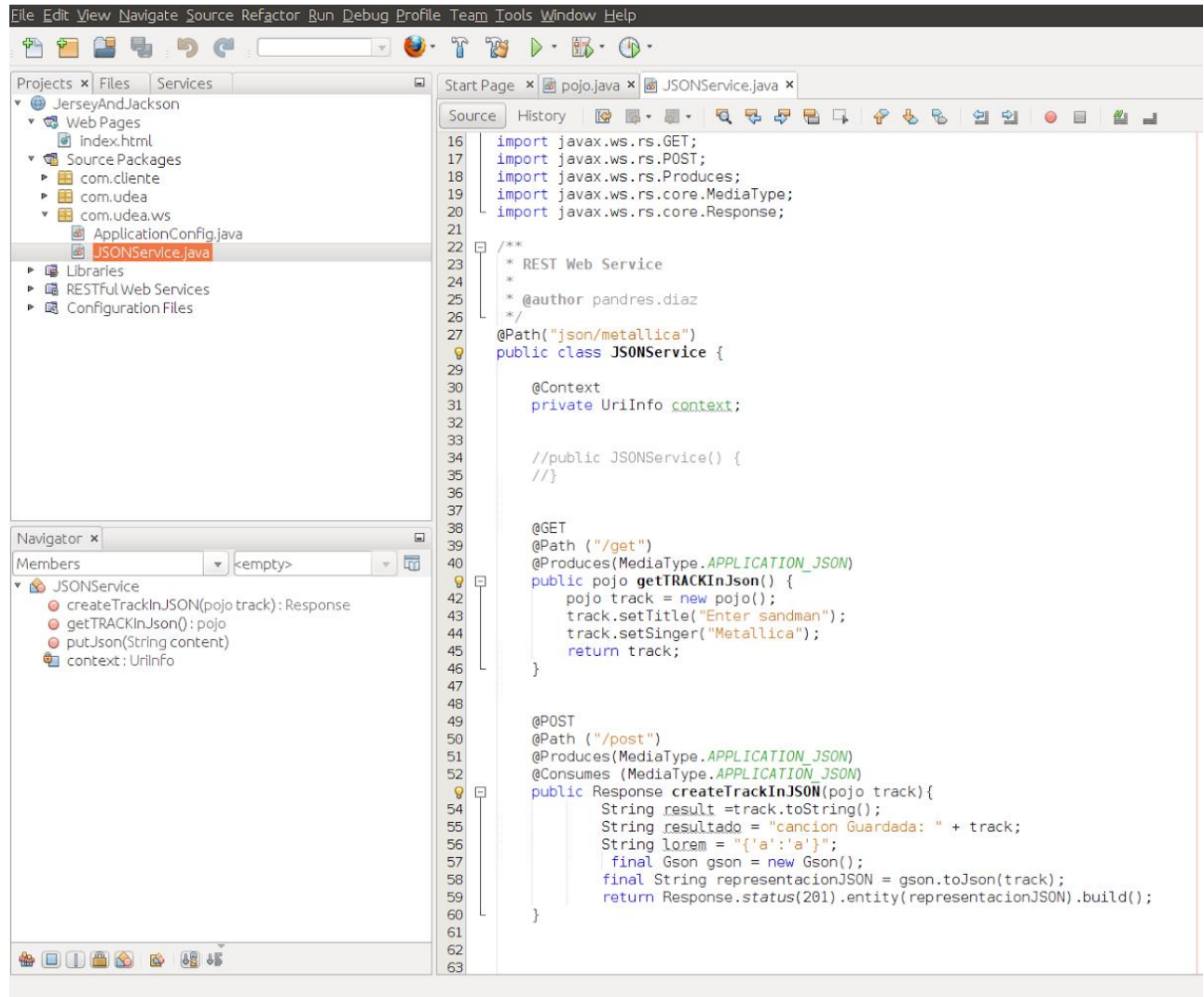
Continuamos creando el package “com.udea” y dentro de este una clase llamada “pojo”



Procedemos a crear un nuevo servicio web RESTful tal y como se presenta en el documento guía, teniendo en cuenta de seleccionar el RESTful Web Services, como se muestra en la guía.

Como podemos apreciar el Path es /json/metallica permitiéndonos hacer referencia del recurso directamente desde la URL, y con tipo MIME application/json

Los métodos que aparecen por defecto serán editados de la siguiente manera:



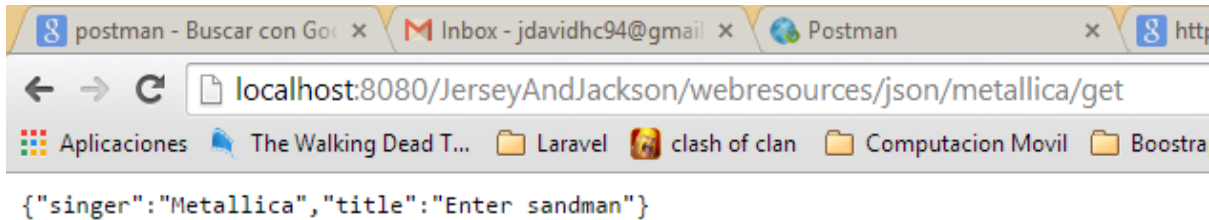
The screenshot shows an IDE with the following components:

- Projects View:** Shows a project named 'JerseyAndJackson' with a 'Web Pages' folder containing 'index.html'. Under 'Source Packages', there is a package 'com.udea.ws' containing 'ApplicationConfig.java' and 'JSONService.java'.
- Navigator View:** Shows the 'JSONService' class with methods: 'createTrackInJSON(pojo track): Response', 'getTRACKInJson(): pojo', 'putJson(String content)', and 'context: UriInfo'.
- Source View:** Displays the code for 'JSONService.java'. The code includes imports for RESTful web service annotations, a class comment, and two methods: 'getTRACKInJson()' and 'createTrackInJSON()'.

```
16 import javax.ws.rs.GET;
17 import javax.ws.rs.POST;
18 import javax.ws.rs.Produces;
19 import javax.ws.rs.core.MediaType;
20 import javax.ws.rs.core.Response;
21
22 /**
23  * REST Web Service
24  *
25  * @author pandres.diaz
26  */
27 @Path("/json/metallica")
28 public class JSONService {
29
30     @Context
31     private UriInfo context;
32
33     //public JSONService() {
34     //}
35
36
37
38     @GET
39     @Path("/get")
40     @Produces(MediaType.APPLICATION_JSON)
41     public pojo getTRACKInJson() {
42         pojo track = new pojo();
43         track.setTitle("Enter sandman");
44         track.setSinger("Metallica");
45         return track;
46     }
47
48
49     @POST
50     @Path("/post")
51     @Produces(MediaType.APPLICATION_JSON)
52     @Consumes(MediaType.APPLICATION_JSON)
53     public Response createTrackInJSON(pojo track){
54         String result = track.toString();
55         String resultado = "cancion Guardada: " + track;
56         String lorem = "{ 'a': 'a' }";
57         final Gson gson = new Gson();
58         final String representacionJSON = gson.toJson(track);
59         return Response.status(201).entity(representacionJSON).build();
60     }
61
62
63 }
```

Procedemos a hacer un llamado al método getTrackinJson, que se utiliza para la conversión automática del archivo, conteniendo el título y el cantante de la canción.

Luego se hace un llamado a un cliente RESTFul Jersey donde se va a crear el cliente que va a consumir los servicios creados como esta en la guía suministrada por el profesor, al entrar al link de servicio en el navegador obtenemos el siguiente resultado.



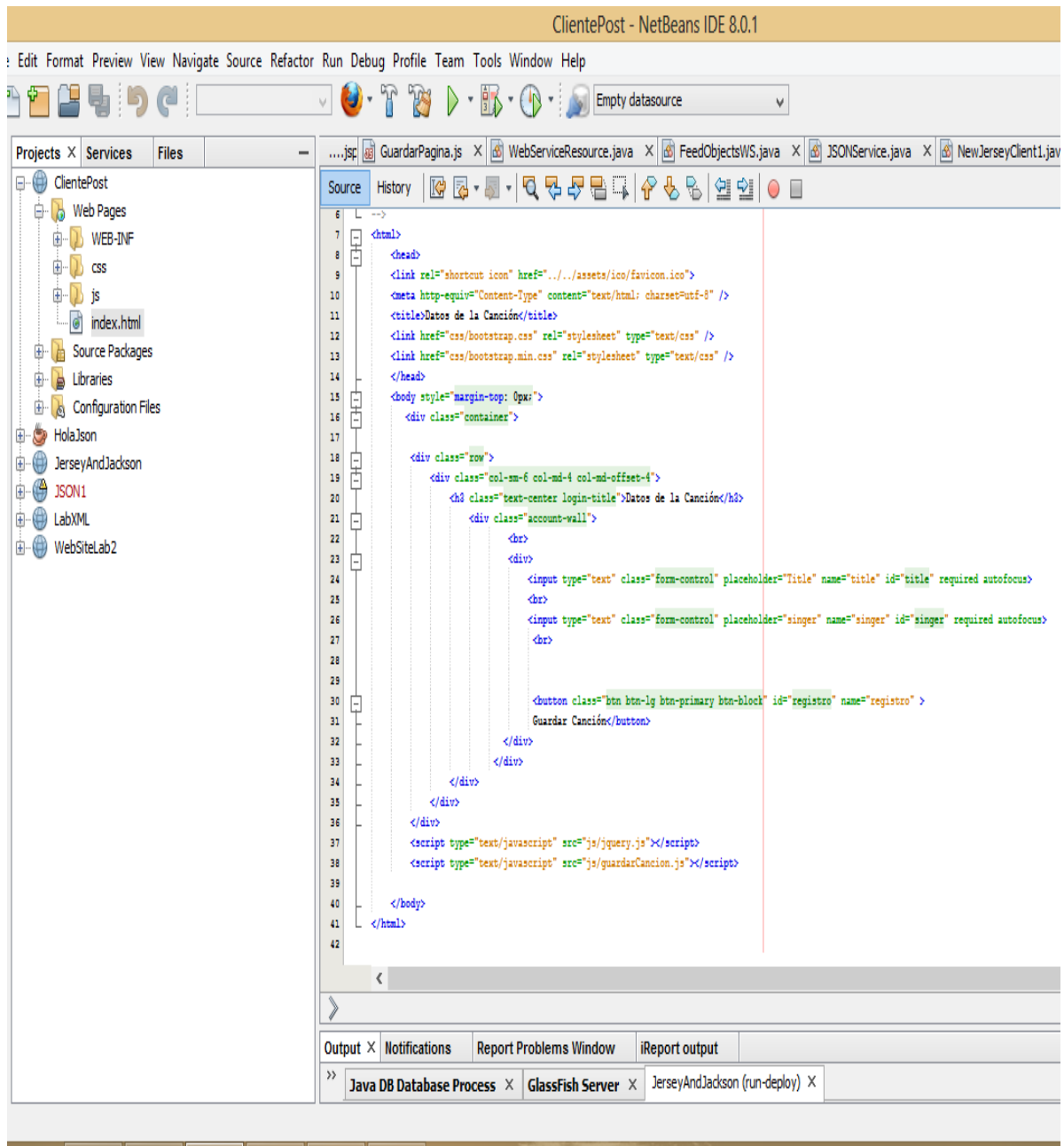
Esta imagen nos asegura que el servicio está funcionando correctamente ya que como respuesta obtenemos un objeto json con la información de una canción.

Ejercicio:

Para probar el request POST, deberá crear un cliente java que permitirá consumir el JSON al patrón URI “/json/metallica/post”. Se pide hacer un programa JAVA que facilita esta tarea.

Ahora procedemos a probar el método POST del servicio Web, creando un nuevo cliente java para poder convertir del formato JSON en el patrón pedido URI

La siguiente imagen nos ilustra cómo se crea el index.html que nos permitirá consumir el servicio pedido.



Después de realizar el index.html procedemos a crear el javascript que es el encargado de hacer el llamado mediante ajax al servicio web, se le pasa como parámetros los datos de la canción que se desea almacenar, si el llamado al servicio es exitoso y se registra la canción se le muestra un mensaje al usuario, informando que la canción se registró exitosamente.

```
...a.js WebServiceResource.java X FeedObjectsWS.java X JSONService.java X NewJerseyClient1.java
Source History
1 $(window).ready(function () {
2     $("#registro").click(function () {
3         registrar();
4     });
5 });
6
7 function registrar(){
8
9     datos = {
10         title: $("#title").val(),
11         singer: $("#singer").val()
12     };
13
14     $.ajax({
15         url: "http://localhost:8080/JerseyAndJackson/webresources/json/metallica/post",
16         type: "POST",
17         contentType: "application/json",
18         data: JSON.stringify(datos),
19         dataType: "json",
20         success: function(json, b, c){
21             console.log(json);
22             console.log(b);
23             console.log(c);
24             if (json){
25                 alert("Felicitaciones! Se guardo la canción correctamente!");
26                 alert("La canción de titulo "+json.title+" del Autor "+json.singer+" se ha Guardado Correctamente");
27             }
28             else {
29                 console.log("Failed: " + json.title + " - " + json.singer);
30                 alert("Ha ocurrido un error guardando la canción. Intente más tarde.");
31             }
32         },
33         error: function(json, b, c) {
34             console.log(json);
35             console.log(b);
36             console.log(c);
37             alert("Ha ocurrido un error enviando el formulario de registro. Intente más tarde.");
38         }
39     });
40     // completa: function(a, b, c) {
```

Al Correr nuestra aplicación nos aparece el siguiente formulario, en el cual debemos ingresar la información de la canción que deseamos almacenar, luego presionamos clic en el botón “Guardar Canción” con lo cual se realiza la petición al servicio



The screenshot shows a web browser window with the title 'Datos de la Canción'. The address bar shows 'localhost:8080/ClientePost/'. The page has a title 'Datos de la Canción'. There are two input fields: the first contains 'Undisclose Desire' and the second contains 'Muse'. Below the fields is a blue button labeled 'Guardar Canción'. The browser's taskbar at the bottom shows various icons and the system clock indicating 2:46 p.m. on 07/11/2014.

A Continuación este es el mensaje del servicio creado, el cual se consumió correctamente, nos muestra un mensaje informando que la canción se almacenó correctamente



The screenshot shows the same web browser window as before, but with a confirmation dialog box displayed in the center. The dialog box has the text: 'La cancion de titulo undisclosed desire del Autor Muse se ha Guardado Correctamente'. Below this text is a checkbox labeled 'Evitar que esta página cree diálogos adicionales'. At the bottom right of the dialog box is an 'Aceptar' button. The background of the browser window is dimmed. The system clock at the bottom right now shows 5:25 p.m. on 06/11/2014.

3. Consumo de una BD MySQL hacia JSON

Se procederá a crear una BD para registrar las páginas web más empleadas por un usuario y posteriormente se hará su integración con JSON por medio de un servicio web RESTful.

Después de seguir la guía suministrada por el profesor, creamos una clase de tipo Web Services RESTful API, tal como se realizó en el ejercicio 2, quedando de la siguiente manera.

En ella están creados los métodos GET y PUT del servicio Web

A Continuación veremos el código del método GET y su funcionamiento

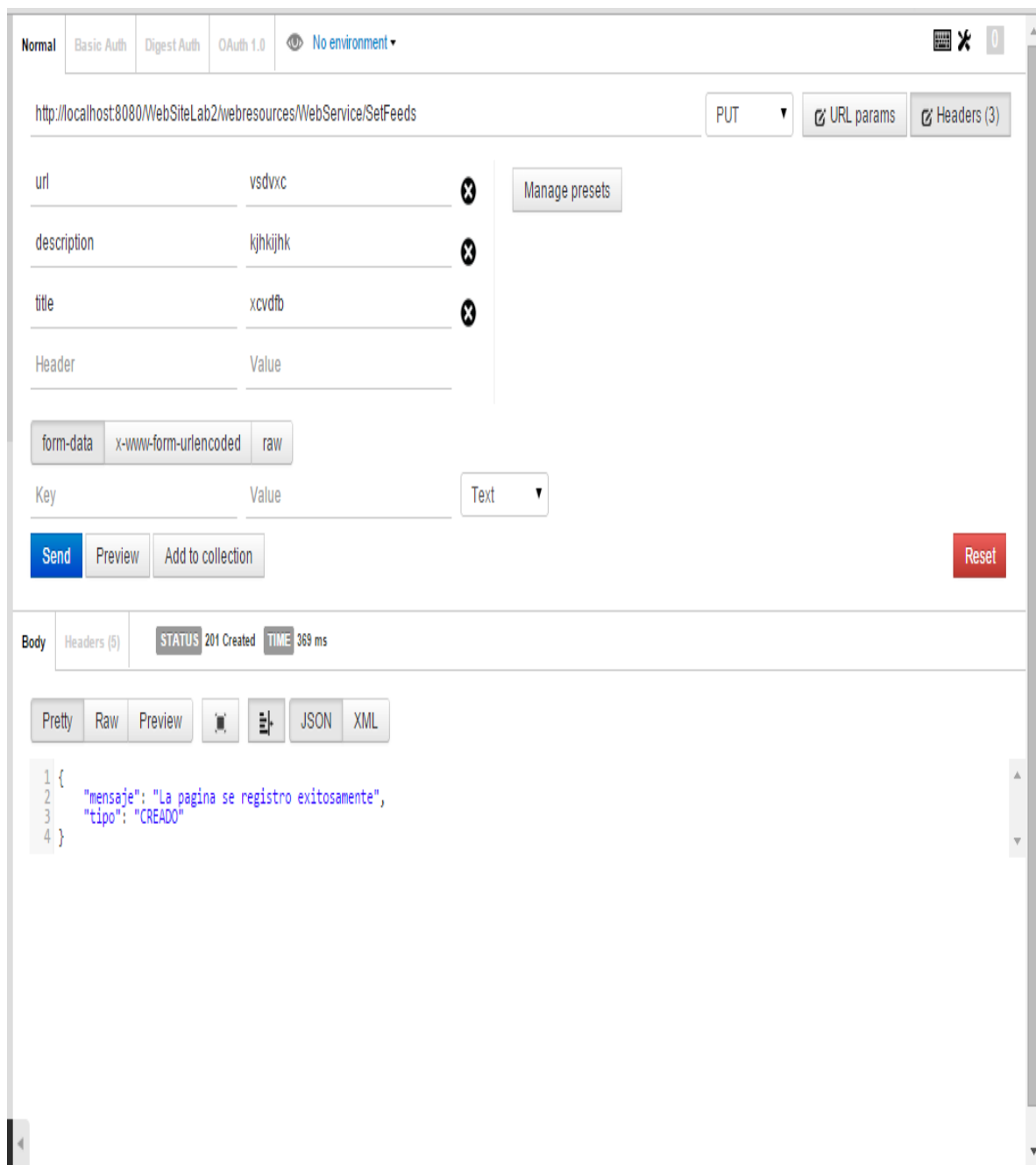
```
@Path("WebService")
public class WebServiceResource {

    @Context
    private UriInfo context;

    @GET
    @Path("/GetFeeds")
    @Produces(MediaType.APPLICATION_JSON)
    public String feed() {
        System.out.println("Entro al servicio");
        String feeds = "";
        try {
            ArrayList<FeedObjects> feedData = new ArrayList<>();
            ProjectManager projectManager = new ProjectManager();
            feedData = projectManager.GetFeeds();
            System.out.println("Paso");
            Gson gson = new Gson();
            System.out.println(gson.toJson(feedData));
            feeds = (gson.toJson(feedData));
        } catch (Exception e) {
            System.out.println("Exception Error");
        }
        return feeds;
    }
}
```

Después de esto se procede a desplegar la aplicación por medio de glassfish y abrimos el navegador para colocar la siguiente URL

<http://localhost:8080/RESTfulProject/webresources/WebService/GetFeeds> qué es la correspondiente al método GET del servicio web, el resultado al probar el método por medio de Postman es el siguiente.



Ahora veremos el código del método PUT y su funcionamiento, este método recibe la información de una página web, es decir, el título, la descripción y la url y procede a almacenarla en la DB para luego retorna un objeto json con los parámetros tipoMensaje y Mensaje en los cuales se informa si se almaceno correctamente la página web y el mensaje que se mostrara al usuario


```

@PUT
@Path("/SetFeeds")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public Response InsertarPagina(@HeaderParam("title") String title,
                               @HeaderParam("description") String description,
                               @HeaderParam("url") String url) throws Exception {

    Respuesta respuesta = new Respuesta();
    ProjectManager managerPagina = new ProjectManager();
    String representacionJSON = "";
    FeedObjects pg = new FeedObjects();

    if ("".equals(title) || "".equals(description) || "".equals(url) || title == null
        || description == null || url == null) {
        respuesta.setTipo(Respuesta.FALTA_CAMPOS);
        respuesta.setMensaje("Ingresa todos los campos de la petición");
        return Response.status(Response.Status.BAD_REQUEST)
            .type(MediaType.APPLICATION_JSON)
            .entity(respuesta)
            .build();
    }

    pg.setTitle(title);
    pg.setDescription(description);
    pg.setUrl(url);
    try {
        int result = managerPagina.setFeeds(pg);
        if (result == 1) {
            final Geon geon = new Geon();
            representacionJSON = geon.toJson(pg);
        }
    } catch (Exception e) {
        respuesta.setTipo(Respuesta.ERROR);
        respuesta.setMensaje("Ocurrió un error registrando la pagina. Intenta luego");
        return Response.status(Response.Status.CONFLICT)
            .type(MediaType.APPLICATION_JSON)
            .entity(respuesta)
            .build();
    }

    respuesta.setTipo(Respuesta.CREADO);
    respuesta.setMensaje("La pagina se registro exitosamente");
    return Response.status(Response.Status.CREATED)
        .type(MediaType.APPLICATION_JSON)
        .entity(respuesta)
        .build();
}

```

Normal
Basic Auth
Digest Auth
OAuth 1.0
No environment

http://localhost:8080/WebSiteLab2/webresources/WebService/SetFeeds
PUT
URL params
Headers (3)

url	vsvdvc	✕
description	kjhkijhk	✕
title	xcvdfb	✕
Header	Value	

form-data
x-www-form-urlencoded
raw

Key
Value
Text

Send
Preview
Add to collection
Reset

Body
Headers (5)
STATUS 201 Created
TIME 369 ms

Pretty
Raw
Preview
JSON
XML

```

1 {
2   "mensaje": "La pagina se registro exitosamente",
3   "tipo": "CREADO"
4 }

```

Luego de ver internamente el funcionamiento de los métodos del servicio web procedemos a realizar Deploy de la aplicación y luego Run

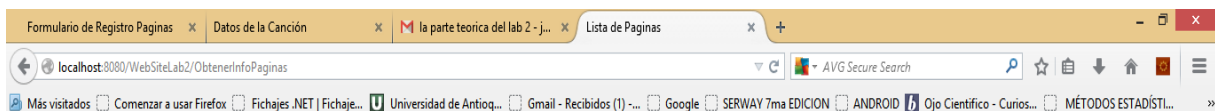
La página del Index de la aplicación es la siguiente, en ella podemos probar los dos métodos del servicio web, si presionamos en el botón “Listar Pagina”, se traerá mediante el método GET una lista con la información de las páginas que tenemos en la DB, si presionamos el botón “Ingresar Pagina” nos llevara a otro formulario en el cual debemos ingresar los datos de la página que deseamos almacenar.

Obtener Información Paginas Web

Listar Paginas

Ingresar Pagina

A continuación veremos el resultado de presionar en el botón “Listar Paginas”



Paginas Registradas

Title	Description	Uri
wikipedia	Enciclopedia	www.wikipedia.com
Amazon	Pagina para el comercio	www.amazon.com
Google	Pagina de Búsqueda	www.google.com

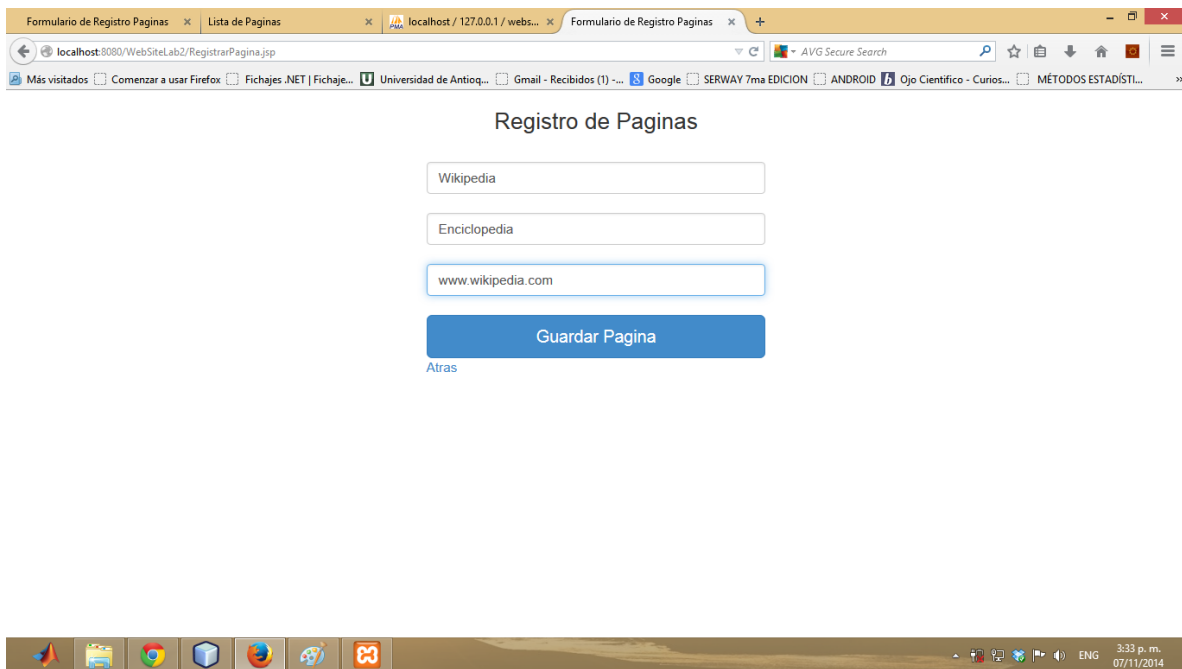
Si tenemos paginas registradas en la DB están serán mostradas en una lista, si se produce un error al acceder a la base de datos mediante el servicio web entonces se mostrara un mensaje informando el error ocurrido.



Si no existen páginas web registradas en la DB, se muestra un mensaje de notificación al usuario



Ahora procedemos a probar el método PUT del servicio ingresando una nueva página web



Formulario de Registro Paginas x Lista de Paginas x localhost / 127.0.0.1 / webs... x Formulario de Registro Paginas x +

localhost:8080/WebSiteLab2/RegistrarPagina.jsp

Más visitados Comenzar a usar Firefox Fichajes .NET | Fichaje... Universidad de Antioq... Gmail - Recibidos (1) -... Google SERWAY 7ma EDICION ANDROID Ojo Científico - Curios... MÉTODOS ESTADÍSTI...

Registro de Paginas

Wikipedia

Enciclopedia

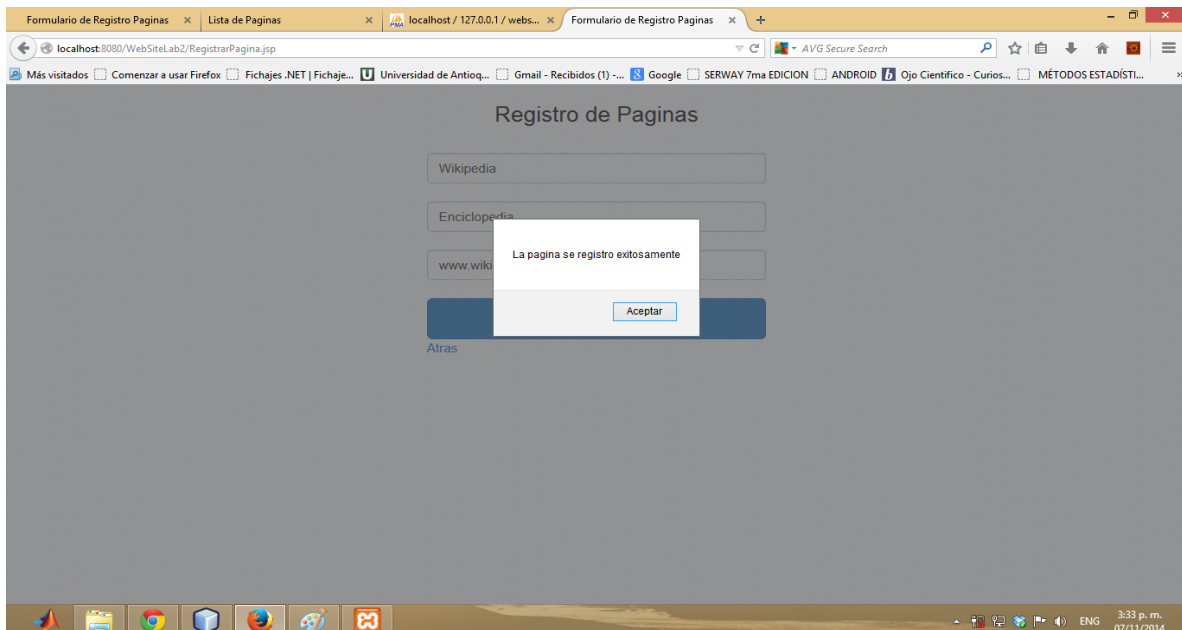
www.wikipedia.com

Guardar Pagina

Atras

3:33 p. m. 07/11/2014

Si el almacenamiento de la página se realiza correctamente, este será el mensaje que se le mostrara al usuario, luego de esto se puede ir de nuevo a Listar las paginas existentes e inmediatamente notara que la página que registro ya se encuentra en la DB



Formulario de Registro Paginas x Lista de Paginas x localhost / 127.0.0.1 / webs... x Formulario de Registro Paginas x +

localhost:8080/WebSiteLab2/RegistrarPagina.jsp

Más visitados Comenzar a usar Firefox Fichajes .NET | Fichaje... Universidad de Antioq... Gmail - Recibidos (1) -... Google SERWAY 7ma EDICION ANDROID Ojo Científico - Curios... MÉTODOS ESTADÍSTI...

Registro de Paginas

Wikipedia

Enciclopedia

www.wiki

La pagina se registro exitosamente

Aceptar

Atras

3:33 p. m. 07/11/2014

Conclusiones

- La utilización de json permite una rápida y eficiente transferencia de información entre un cliente y un servidor.
- Netbeans nos brinda a los programadores herramientas que pueden agilizar la construcción de servicios web.
- Jersey y Jackson facilitan el parseo de objetos java a json y el proceso inverso lo que le permite al desarrollador realizar rápidamente esta tarea
- La implementación de servicios web permite comunicar el frontend y backend de una aplicación y mediante las herramientas que ofrece netbeans este proceso se puede realizar más rápido.
- Mediante los métodos GET, POST,PUT y DELETE podemos realizar todas las operaciones de CRUD de nuestra DB.

Referencias

- <https://jersey.java.net/apidocs/latest/jersey/index.html>
- <http://www.java2s.com/Code/Jar/j/Downloadjacksonall190jar.htm>
- <http://www.programacion.com.py/web/java-web/web-service-rest-jersey-con-java-ee>
- <http://es.wikipedia.org/wiki/JSON>
- <http://www.aprenderaprogramar.com.co/Cursos/Java-Avanzado/RESTful/>
- <http://api.jquery.com/jquery.ajax/>
- <http://stackoverflow.com/questions/12347211/pass-accepts-header-parameter-to-jquery-ajax>

Información:

El código de todos y cada uno de los proyectos que se muestran en este documento estarán adjuntados en la carpeta de la entrega final que se dará al profesor, para que puedan ser probados.