Trabajo de Investigación

universidad nacional de ingeniería | Sistemas UNI

Manejo de Datos en Formato JSON

aLUMNO: Jose Alejandro Chavez Rebatta

2020

Contenido

[Dedicatoria 2](#_Toc34480390)

[1. Resumen 3](#_Toc34480391)

[2. Desarrollo teórico 3](#_Toc34480392)

[2.1. ¿Qué es JSON? 3](#_Toc34480393)

[2.2. Lenguajes de Programación 3](#_Toc34480394)

[2.3. Sintaxis 1](#_Toc34480395)

[2.3.1. Objeto 1](#_Toc34480396)

[2.3.2. Arreglo 1](#_Toc34480397)

[2.3.3. Valor 1](#_Toc34480398)

[2.4. JSON vs XML 2](#_Toc34480399)

[3. Aplicación 3](#_Toc34480400)

[3.1. Proceso de instalación de GSON 3](#_Toc34480401)

[3.2. Serializar 4](#_Toc34480402)

[3.3. Deserializar 7](#_Toc34480403)

[4. Conclusiones 10](#_Toc34480404)

[5. Recomendaciones 10](#_Toc34480405)

[6. Bibliografía 11](#_Toc34480406)

# Dedicatoria

Este presente trabajo está dedicado a todas las personas que me han apoyado para llevar este curso y espera que triunfe y me supere cada día

# Resumen

El presente trabajo de investigación tiene como objetivo principal dar a conocer las características de usar información en formato Json.

Asimismo, se desarrollaron 2 casos aplicativos basados en la explicación teórica:

* Una aplicación que permite convertir información en Java a un objeto Json, así como guardar dicha información en un archivo.
* Una aplicación que permite extraer información específica de un objeto JSON en Java.

Ambos casos se desarrollaron en el IDE Netbeans y empleando la librería GSON

**Palabras claves:** Deserializar,Gson, Json, Netbeans, Serializar

# Desarrollo teórico

## ¿Qué es JSON?

[JSON](http://json.org/) (JavaScript Object Notation) es un formato de texto pensado para el intercambio de datos. El formato JSON fue definido por Douglas Crockford a finales de 2002 y dado a conocer a través de su página web <http://json.org/>. Esta página contenía la definición del formato y una implementación en Java. En poco tiempo el uso del formato se extendió y aparecieron implementaciones para todos los lenguajes de programación.

|  |  |
| --- | --- |
| **JSON** | |
| **Es:** | **No es:** |
| 1. Formato ligero. | 1. Difícil de manejar |
| 1. Basado en texto. | 1. Formato de archivo |
| 1. Sencillo y de fácil uso. | 1. Lenguaje de programación |
| 1. Fácil de parsear. | 1. Extensible |
| 1. Independiente de cualquier lenguaje de programación. | 1. Lenguaje de marcado |

## Lenguajes de Programación

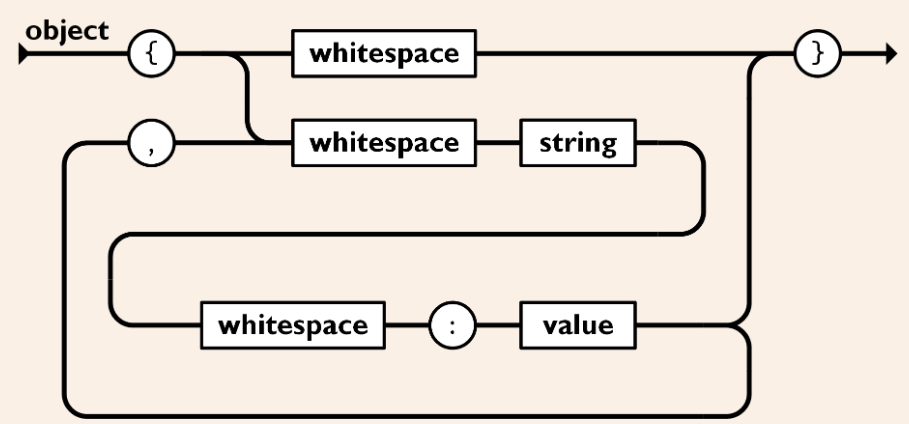
JSON es compatible con todos los lenguajes de programación entre los cuales podemos mencionar:

* .NET (C#,VB.NET…)
* C
* C+´+
* Java
* JavaScript
* Perl
* PHP
* Python
* Rubí
* Etc

## Sintaxis

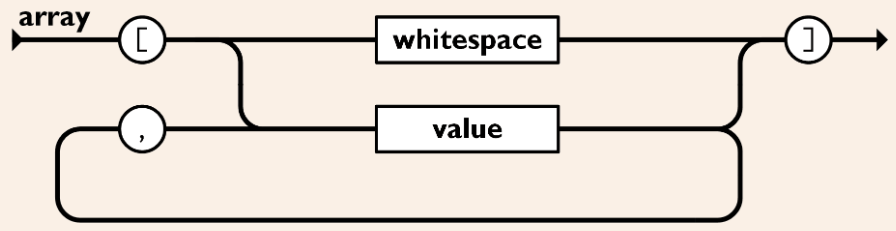
### Objeto

* Es un conjunto de pares nombre/valor.
* Está contenido en **llaves {}.**
* Los **nombres** deben ser **cadenas** y los **valores** deben ser un **tipo de datos JSON** válido.
* Los nombres y los valores están separados por **dos puntos**
* Cada **par nombre / valor** está separado por una **coma.**



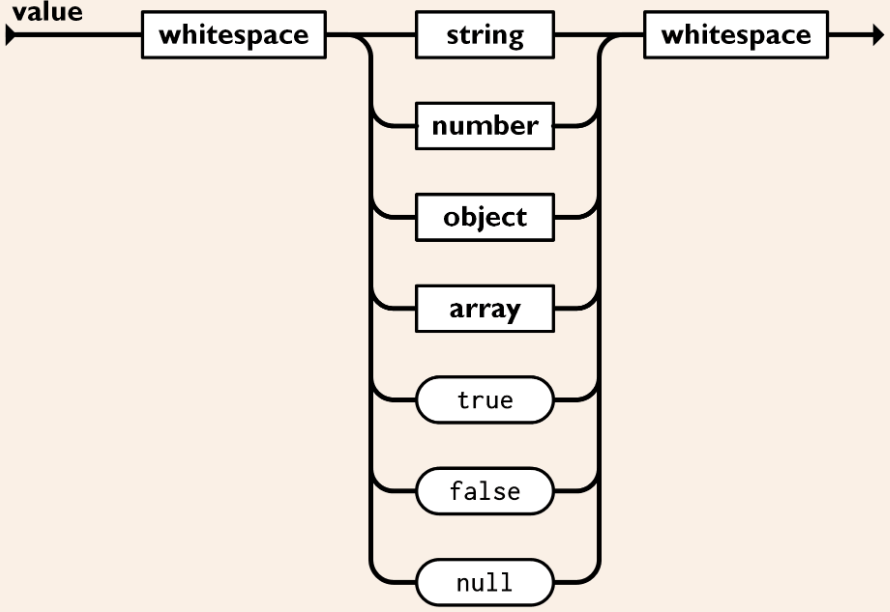
### Arreglo

* Colección de valores.
* Está contenido en **corchetes [ ].**
* Los **nombres** deben ser **cadenas** y los **valores** deben ser un **tipo de datos JSON** válido



### Valor

* Puede ser de tipo **cadena, número, objeto, matriz, booleano o nulo**.
* Si es una **cadena de caracteres** debe estar entre **comillas “”.**
* Pueden a**nidarse.**



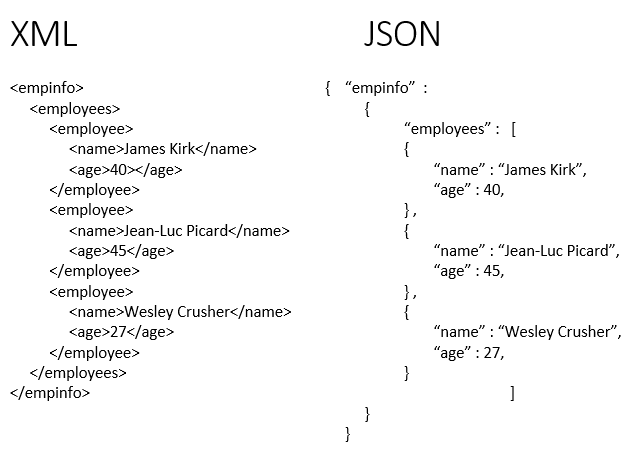
## JSON vs XML

**XML:**

XML es un lenguaje de marcado, el cual su acrónimo se entiende por Extensible Markup Language (Lenguaje de Marcado Extensible) y es un lenguaje de marcado de propósito general. A diferencia de otros lenguajes de marcado, XML no está predefinido, por lo que se debe y permite definir etiquetas propias. El principal propósito de este lenguaje es compartir datos a través de diferentes sistemas, como, por ejemplo Internet.

|  |  |
| --- | --- |
| **XML VS JSON** | |
| **JSON** | **XML** |
| 1. Basado en JavaScript | 1. Basado en etiquetas |
| 1. No extensible | 2. Extensible |
| 1. Son datos sin significados | 3. Datos sin significados |
| 1. Soporta datos estructurados en arreglos | 4. No soporte arreglos |
| 1. Más ligero y rápido | 5. Más pesado y lento |

Ejemplo:



Fuente: QUORA

# Aplicación

A continuación, se un empleo práctico de

Para este ejercicio se empleará el

## Proceso de instalación de GSON

|  |
| --- |
| 1. Ingresar al siguiente link: <http://www.java2s.com/Code/Jar/g/Downloadgson222jar.htm> y descargar el comprimido. |
| **Imagen 1: Descarga de GSON** |
| 1. Hacer clic derecho a proyecto en el que se va emplear y seleccionar “Propiedades” |
| 1. En la pantalla de Propiedades, en la parte de Categorías, seleccionar “**Librerías”** y hacer clic en el botón **“Añadir JAR/Folder “** |
|  |

## Serializar

El objetivo es crear una aplicación que permita convertir en formato JSON (objeto), la información referente a la encuesta para profesores en donde se considera los datos del profesor como el Nombre, apellido paterno, apellido materno, cantidad de alumnos, nota, y condición de aprobado desaprobado.

Asimismo, contará con tres botones los cuales permitirán las siguientes acciones:

* **Convertir**: Convertir información en JAVA a un objeto JSON, así como guardar la información en un archivo JSON.
* **Borrar**: Limpiar los campos solicitados.
* **Salir**: Cerrar la aplicación:

|  |
| --- |
| **Aplicación: Serializar** |
|  |
|  |
| 1. Se crea la Clase Model donde se declara los **get** and **set** de las variables, así como los constructores respectivos. |
| **Clase Model**  package es.com.serializar.model;  public class Model {  private String Nombre;  private String APaterno;  private String AMaterno;  private int CantAlumnos;  private float Nota;  private boolean Aprobado;    public Model(String Nombre, String APaterno, String AMaterno, int CantAlumnos, float Nota, boolean Aprobado) {  this.Nombre = Nombre;  this.APaterno = APaterno;  this.AMaterno = AMaterno;  this.CantAlumnos = CantAlumnos;  this.Nota = Nota;  this.Aprobado = Aprobado;    }  public Model() {    }  public String getNombre() {  return Nombre;  }  public void setNombre(String Nombre) {  this.Nombre = Nombre;  }  public String getAPaterno() {  return APaterno;  }  public void setAPaterno(String APaterno) {  this.APaterno = APaterno;  }  public String getAMaterno() {  return AMaterno;  }  public void setAMaterno(String AMaterno) {  this.AMaterno = AMaterno;  }  public int getCantAlumnos() {  return CantAlumnos;  }  public void setCantAlumnos(int CantAlumnos) {  this.CantAlumnos = CantAlumnos;  }  public float getNota() {  return Nota;  }  public void setNota(float Nota) {  this.Nota = Nota;  }  public boolean isAprobado() {  return Aprobado;  }  public void setAprobado(boolean Aprobado) {  this.Aprobado = Aprobado;  }    } |
| 1. En la Clase View, se genera el código para obtener la información correspondiente y generar el archivo |
| private void btnConvertActionPerformed(java.awt.event.ActionEvent evt) {  **//Previas**  btnConvert.setEnabled(false);  btnDelete.setEnabled(true);    **//Variables**  Model model = new Model();  model.setNombre(txtNombre.getText());  model.setAPaterno(txtAPaterno.getText());  model.setAMaterno(txtAMaterno.getText());  model.setCantAlumnos(Integer.parseInt(txtCantAlumnos.getText()));  model.setNota(Float.parseFloat(txtNota.getText()));  model.setAprobado(Boolean.parseBoolean(txtAprobado.getText()));  **//Proceso**  Gson gson = new Gson();  String json =gson.toJson(model);  **//Guardar archivo**  try (FileWriter file = new FileWriter("file1.json")) {  gson.toJson(model);  file.write(json);  file.flush();  } catch (IOException e) {  }    **//Resultado**  txtSalida.setText("" + json);    **}** |

## Deserializar

A diferencia que el caso anterior, esta vez se crea una aplicación la cual permite obtener información específica a partir de un objeto JSON

|  |
| --- |
| **Aplicación: Deserializar** |
|  |
|  |
| 1. Se crea la Clase Model donde se declara los **get** and **set** de las variables, así como los constructores respectivos. |
| **Clase Model**  package es.com.deserializar.model;  public class Model {  //Entrada  private String Cadena;  //Salida  private String Nombre;  private String APaterno;  private String AMaterno;  private int CantAlumnos;  private float Nota;  private boolean Aprobado;    public Model() {    }  public Model(String Cadena) {  this.Cadena = Cadena;  }  public String getCadena() {  return Cadena;  }  public void setCadena(String Cadena) {  this.Cadena = Cadena;  }  public String getNombre() {  return Nombre;  }  public void setNombre(String Nombre) {  this.Nombre = Nombre;  }  public String getAPaterno() {  return APaterno;  }  public void setAPaterno(String APaterno) {  this.APaterno = APaterno;  }  public String getAMaterno() {  return AMaterno;  }  public void setAMaterno(String AMaterno) {  this.AMaterno = AMaterno;  }  public int getCantAlumnos() {  return CantAlumnos;  }  public void setCantAlumnos(int CantAlumnos) {  this.CantAlumnos = CantAlumnos;  }  public float getNota() {  return Nota;  }  public void setNota(float Nota) {  this.Nota = Nota;  }  public boolean isAprobado() {  return Aprobado;  }  public void setAprobado(boolean Aprobado) {  this.Aprobado = Aprobado;  }    } |
| 1. En la Clase View, se genera el código para obtener la información correspondiente y generar el archivo |
| private void btnExtActionPerformed(java.awt.event.ActionEvent evt) {  //Previas  btnExt.setEnabled(false);  btnDelete.setEnabled(true);  //Variable  Model model = new Model();  model.setCadena(txtCadena.getText());  //Proceso  Gson gson = new Gson();  Model c2 = gson.fromJson(txtCadena.getText(),Model.class);  //Result  txtNombre.setText("" + c2.getNombre());  txtAPaterno.setText("" + c2.getAPaterno());  txtAMaterno.setText("" + c2.getAMaterno());  txtCantAlumnos.setText("" + c2.getCantAlumnos());  txtNota.setText("" + c2.getNota());  txtAprobado.setText("" + c2.isAprobado());  **}** |

# Conclusiones

* Se ha demostrado como JSON permite el intercambio de información de una forma más fácil y rápida.
* GSON no es la única librería que facilita el manejo de objetos JSON: Jackson, JSON Simple, JSONP
* Muy usado en aplicaciones web

# Recomendaciones

* Dependiendo del volumen de información, se debe elegir la librería correspondiente
* Adaptar la programación para el uso de Arreglos y listas List <Array>
* Investigar la interacción y el uso de JSON con AJAX
* Tener en cuenta en otros lenguajes de programación, IDE´s

# Bibliografía

1. Sintes, B. (2002). JSON. Recuperado de <https://www.mclibre.org/consultar/informatica/lecciones/formato-json.html>
2. Refsnes Data. (2020). JavaScript Tutorial. Recuperado de <https://www.w3schools.com/js/>
3. Emmerson Miranda. (2010). Manual de JSON. Formatos ligeros de intercambio de datos en aplicaciones web. Recuperado de <https://es.slideshare.net/Emmerson_Miranda/json-short-manual-5732305>
4. Fox, J. (2013). JSON: The Basics. Recuperado de <https://es.slideshare.net/jfox015/json-22195683>
5. Crockford, D. (2020). Introducción a JSON. How JavaScript Works. Recuperado de <https://www.json.org/json-es.html>