
	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos				CURSO:	2º	
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

Tema 3: JSON

Índice

1. JSON	1
2. Elementos que componen un documento JSON	1
2.1. Datos	1
2.2. Tipos de valores	1
2.3. Objetos JSON	2
2.4. Arrays JSON	2
2.5. Resumen	2
2.6. JSON y XML	3
2.7. Usos de JSON	3
2.8. Herramientas	3
3. JSON y Java	4
3.1. Crear un árbol JSON	6
3.2. Escribir un objeto JSON a un flujo	7
3.3. Navegar por un documento JSON	8
3.4. Creación de un modelo de objetos de código de aplicación	10
4. APIS de servicios web	14
4.1. Open Weather Map	14
4.2. Evenful	15
4.3. Last.fm	16
4.4. Wikipedia	16
4.5. IMDB	17
5. Apéndice 1: API de Google	18
6. Apéndice 2: Evitar bloqueo de peticiones	19
7. Bibliografía	20

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

1. JSON

JSON (JavaScript Object Notation) es una alternativa, más sencilla de usar, a XML para el intercambio de datos.

Nació como un subconjunto de la notación de objetos de JavaScript. Debido a esto un programa JavaScript puede utilizar las funciones estándar, de JavaScript, para convertir datos JSON en objetos nativos de JavaScript, sin tener que utilizar un programa de análisis (como hace XML).

Las similitudes entre JSON y XML son:

- Tanto JSON y XML son texto plano, por lo tanto, legibles para los humanos.
- Tienen una sintaxis muy simple.
- Son jerárquicos (valores dentro de los valores).
- Independientes del lenguaje de programación.

Y sus diferencias son:

- JSON no utiliza etiqueta final.
- JSON es más corto.
- JSON es más rápido de leer y escribir.
- JSON puede utilizar matrices.
- JSON no tiene un esquema ampliamente aceptado para la definición y validación de la estructura de datos JSON.

2. Elementos que componen un documento JSON

2.1. Datos

Los datos se escriben como pares clave:valor.

Un par clave:valor consiste en un nombre de una clave, escrita entre comillas, seguido de dos puntos y un valor.

```
"Nombre": "Juan"
```

Los pares clave:valor se separan entre ellos por comas.

2.2. Tipos de valores

Los tipos de valores que podemos encontrar en JSON son los siguientes:

- Un número.
- Una cadena de cero o más caracteres unicode entre comillas.
- Booleano (verdadero o falso).
- Un array Json (entre corchetes).
- Un objeto Json (entre llaves).
- Null.

JSON representa solamente datos. No define el concepto de variables, asignaciones, igualdades o funciones.

<div><div>COLEXIO</div><div>VIVAS</div><div>S.L.</div></div>	RAMA:	Informática	CICLO:	Desenvolvemeto de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

2.3. Objetos JSON

Los objetos JSON:

- Objetos se escriben entre llaves por lo que empiezan con { y terminan con }.
- Son un conjunto desordenado de pares clave:valor.
- Los pares clave:valor están separados, entre ellos, por comas.
- Las claves, parte izquierda del par, deben ser cadenas y deben ser diferentes entre sí.

Ejemplo:

```
{ "nombre": "Manuel", "Apellido": "Graíño" }
```

2.4. Arrays JSON

Las matrices JSON:

- Están encerradas entre corchetes por lo que una matriz comienza con [y termina con].
- Son una lista ordenada de valores.
- Cada valor puede ser de cualquier tipo, incluyendo objetos o arrays.
- Los valores están separados por comas.

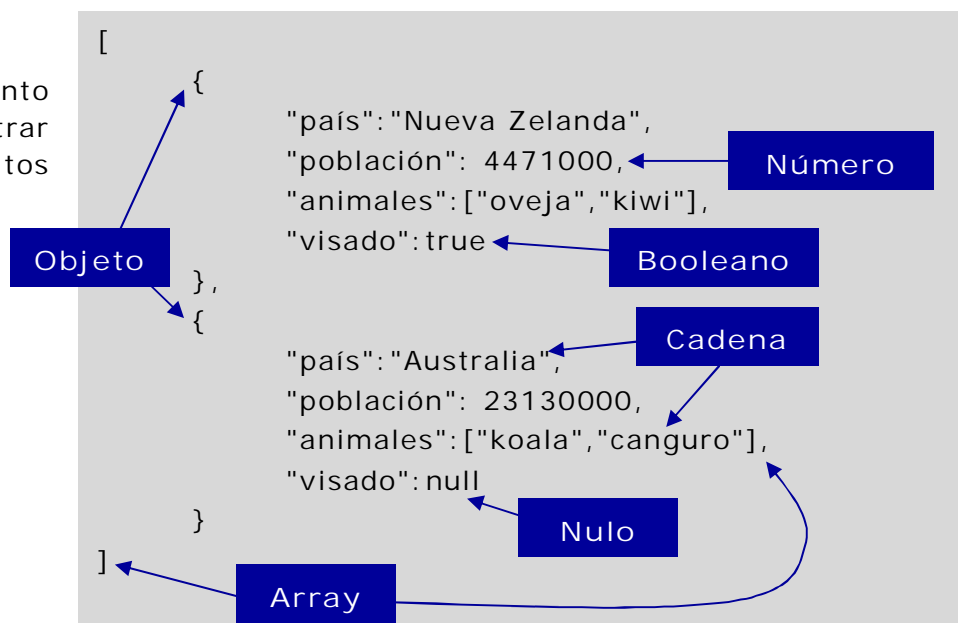
Ejemplo:


```
"empleados": [
  { "nombre": "Manuela", "apellido": "Freire"},
  { "nombre": "Lorenzo", "apellido": "Veiga"},
  { "nombre": "Concepción", "apellido": "Senra"}
]
```

En el ejemplo anterior "empleados" es un array que contiene tres valores que son tres objetos JSON. Cada objeto es un registro de una persona (con un nombre y un apellido).

2.5. Resumen

En el siguiente documento JSON podemos encontrar todos los elementos definidos anteriormente:



	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

2.6. JSON y XML

En este ejemplo JSON se define un objeto empleados, que contiene un array con tres elementos. Cada uno de esos elementos es un objeto que representa un empleado:

```
{ "empleados": [
  { "nombre": "Manuela", "apellido": "Freire"},
  { "nombre": "Lorenzo", "apellido": "Veiga"},
  { "nombre": "Concepción", "apellido": "Senra"},
  { "nombre": "Manuel", "apellido": "Graño"}
]
```

Veamos su equivalente XML:

```
<empleados>
  <empleado >
    <nombre>Manuela</nombre> <apellido>Freire</apellido>
  </empleado>
  <empleado>
    <nombre>Lorenzo</nombre> <apellido>Veiga</apellido>
  </empleado>
  <empleado >
    <nombre>Concepción</nombre> <apellido>Senra</apellido>
  </empleado >
  <empleado>
    <nombre>Manuel</nombre> <apellido>Graño</apellido>
  </empleado>
</empleados>
```

2.7. Usos de JSON

JSON se usa a menudo como un formato común enviar información entre aplicaciones que se comunican entre sí a través de Internet. JSON es independiente de los lenguajes de programación usados para crear estas aplicaciones así como de los entornos donde se ejecuten. Por lo que JSON es adecuado para este escenario, ya que es un estándar abierto, es fácil de leer y escribir, y es más compacto que otras representaciones.


Los servicios web RESTful utilizan ampliamente JSON como el formato de los datos dentro de las solicitudes y respuestas.

2.8. Herramientas

Para visualizar y comprobar la validez de un documento JSON existen varias herramientas online, entre las que podemos encontrar:

- <http://jsonformatter.curiousconcept.com/>
- <http://jsonviewer.stack.hu>
- <http://jsonlint.com/>

Eclipse, NetBeans, Visual Studio Code y Firefox permiten visualizar, de forma nativa, un documento JSON.

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

Notepad++ permite ver y formatear como un árbol un documento JSON. Deberemos instalar el plugin JSTool mediante la herramienta plugin manager que se encuentra en el Menú Plugins.

Una vez instalado podremos usar la opción JSON Viewer situado en el menú Plugins para formatear o ver el árbol de un texto seleccionado.

3. JSON y Java¹

La API JSON de Java nos permite leer, modificar, crear y realizar consultas JSON mediante dos modelos: el modelo de objetos y el modelo de streaming (solo veremos el modelo de objetos).

- El modelo de objetos crea una estructura de árbol en memoria en la se puede navegar y consultar.
- El modelo de streaming ofrece un analizador basado en eventos.

El modelo de objetos de JSON es similar al Modelo de objetos de documento (DOM) de la API XML. Es una API de alto nivel que ofrece los tipos de objetos JsonObject y JsonArray para objetos y matrices JSON.

- JsonObject: ofrece una colección desordenada de cero o más pares de clave:valor.
- JsonArray: ofrece una lista ordenada de cero o más valores.

La siguiente tabla enumera las clases principales y las interfaces en el modelo de objetos.

Clase u interface	Descripción
Json	Contiene métodos estáticos para trabajar con elementos JSON. https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/Json.html https://docs.oracle.com/javaee/7/api/javax/json/Json.html
JsonGenerator	Escribe datos JSON a un flujo https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/stream/JsonGenerator.html https://docs.oracle.com/javaee/7/api/javax/json/stream/JsonGenerator.html
JsonReader	Lee un objeto o array JSON de un flujo de entrada. https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonReader.html https://docs.oracle.com/javaee/7/api/javax/json/JsonReader.html
JsonWriter	Escribe un objeto o array JSON a un flujo de salida. https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonWriter.html https://docs.oracle.com/javaee/7/api/javax/json/JsonWriter.html

¹ Traducción libre de: <https://docs.oracle.com/javaee/7/tutorial/jsonp002.htm>

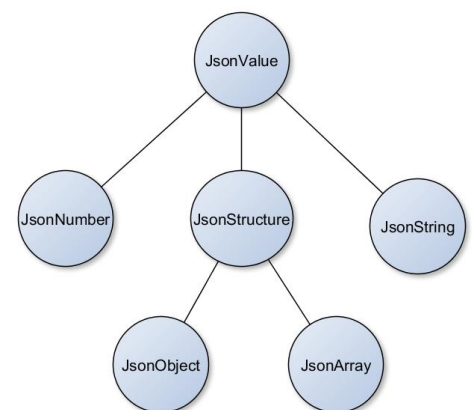
<div>COLEXIO</div> <div>VIVAS S.L.</div>	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

JsonObjectBuilder JsonArrayBuilder	Crean un objeto o una matriz JSON en memoria e proporcionan métodos para añadir pares clave:valor a un objeto o valores a un array. https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonObjectBuilder.html https://docs.oracle.com/javaee/7/api/javax/json/JsonObjectBuilder.html https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonArrayBuilder.html https://docs.oracle.com/javaee/7/api/javax/json/JsonArrayBuilder.html
JsonValue	Representan un valor JSON inmutable. https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonValue.html https://docs.oracle.com/javaee/7/api/javax/json/JsonValue.html
JsonStructure	Supertipo de JsonObject y JsonArrays https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonStructure.html https://docs.oracle.com/javaee/7/api/javax/json/JsonStructure.html
JsonObject JsonArray	Representan un objeto o un array JSON inmutables. Son subtipos de JsonValue. https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonObject.html https://docs.oracle.com/javaee/7/api/javax/json/JsonObject.html https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonArray.html https://docs.oracle.com/javaee/7/api/javax/json/JsonArray.html
JsonString JsonNumber	Representan una cadena JSON y un número JSON inmutables. Son subtipos de JsonValue. https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonString.html https://docs.oracle.com/javaee/7/api/javax/json/JsonString.html https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonNumber.html https://docs.oracle.com/javaee/7/api/javax/json/JsonNumber.html
JsonException	Indica que surgió un problema durante o procesamiento JSON. https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonException.html https://docs.oracle.com/javaee/7/api/javax/json/JsonException.html

Además existen las constantes definidas para los valores JSON: nulo, verdadero y falso.

En los siguientes puntos vamos a describir cuatro casos de uso del modelo de objetos:

- La creación de un modelo de objetos JSON desde un flujo.
- Escribir una modelo de objetos a un flujo.
- La creación un modelo de objetos JSON mediante código.
- La navegación por una modelo de objetos JSON.



La página web oficial de JSON es <https://javaee.github.io/jsonp/> mientras que su documentación se pueden encontrar en la siguiente página web: <https://javadoc.io/static/javax.json/javax.json-api/1.1.4/index.html?overview-summary.html>

Desde la versión 7 de java JSON viene integrado en Java EE (Java Enterprise Edition). Si no disponemos de JAVA EE podemos descargar su jar desde <https://repo1.maven.org/maven2/org/glassfish/javax.json/> o desde <http://www.java2s.com/Code/Jar/j/javax.json.htm> y añadirlo al proyecto.

<div>COLEXIO</div> <div>VIVAS S.L.</div>	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

3.1. Crear un árbol JSON

El código siguiente muestra cómo crear un modelo de objetos de datos JSON desde un archivo de texto o desde una página web ya sea esta http o https:

Ejemplo 1

```
public class Jsonn {
    public JsonValue leeJSON(String ruta) {
        try {
            if (ruta.toLowerCase().startsWith("http://")) {
                return leerHttp(ruta);
            } else if (ruta.toLowerCase().startsWith("https://")) {
                return leerHttps(ruta);
            } else {
                return leerFichero(ruta);
            }
        } catch (IOException e) {
            System.out.println("Error procesando documento Json " +
                               e.getLocalizedMessage());
            return null;
        }
    }

    public JsonValue leerFichero(String ruta) throws FileNotFoundException {
        try (JsonReader reader = Json.createReader(new FileReader(ruta))) {
            return reader.read();
        } /*
        * JsonStructure jsonSt = reader.read();
        * System.out.println(jsonSt.getValueType());

        * JsonObject jsonObj = reader.readObject();
        * System.out.println(jsonObj.getValueType());

        * JsonArray jsonArr = reader.readArray();
        * System.out.println(jsonArr.getValueType());
        */
    }

    public JsonValue leerHttp(String direccion) throws IOException {
        URL url = new URL(direccion);
        try (InputStream is = url.openStream();
             JsonReader reader = Json.createReader(is)) {
            return reader.read();
        }
    }

    public JsonValue leerHttps(String direccion) throws IOException {
        URL url = new URL(direccion);
        HTTPSURLConnection conn = (HTTPSURLConnection) url.openConnection();
        try (InputStream is = conn.getInputStream();
             JsonReader reader = Json.createReader(is)) {
            return reader.read();
        } finally {
            conn.disconnect();
        }
    }
}
```

<div>COLEXIO</div> <div>VIVAS S.L.</div>	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

Dependiendo del contenido del archivo se puede leer un objeto de tipo JsonObject o de tipo JSONArray, los cuales son subtipos de JsonStructure y este a su vez de JsonValue.

Veremos dos ejemplos de la creación de un modelo de datos JSON: uno usando un fichero de texto, llamado países.json, y otro utilizando un servicio web que proporciona información sobre la Formula 1.

Ejemplo 2

```
public String getResultadosPiloto(String piloto, int anho){
    JsonValue json=leeJSON("http://ergast.com/api/f1/" + anho
                           + "/drivers/" + piloto + "/results.json");

    return json.toString();
}

public static void main(String[] args) {
    Jsonn j=new Jsonn();
    File f = new File("d:/países.json");
    JsonValue json=j.leeJSON(f.getAbsolutePath());
    System.out.println(json.toString());

    System.out.println(j.getResultadosPiloto("alonso", 2022));
}
```

3.2. Escribir un objeto JSON a un flujo


Un objeto JSON se puede escribir en un flujo utilizando la clase JsonWriter de la siguiente manera:

Ejemplo 3

```
public void escribeJSON (JsonValue json, File f) throws
    FileNotFoundException{

    System.out.println("Guardando tipo: " + json.getValueType());
    PrintWriter pw=new PrintWriter(f);
    JsonWriter writer=Json.createWriter(pw);

    // writer.write((JsonStructure) json);
    if (json.getValueType() == JsonValue.ValueType.OBJECT) {
        writer.writeObject(json.asJsonObject());
        // writer.writeObject((JsonObject)json);
    } else if (json.getValueType() == JsonValue.ValueType.ARRAY) {
        writer.writeArray(json.asJsonArray());
        // writer.writeArray((JsonArray)json);
    } else System.out.println("No se soporta la escritura");
    writer.close();
}
```


	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

Donde:

- El método `createWriter` crea un `writer` JSON para escribir un objeto o array JSON en el flujo especificado como parámetro (el `PrintWriter` `pw` en el ejemplo).
- Los métodos:
 - `writer`: Mediante el supertipo `JsonStructure` escribe un objeto o array JSON al flujo de escritura.
 - `writeObject`: escribe un objeto JSON en el flujo.
 - `writeArray`: escribe un array JSON en el flujo.
- El método `close` cierra el flujo de escritura.

3.3. Navegar por un documento JSON

Mediante los métodos `get` de `JsonObject` y `JsonArray` podemos obtener los elementos de los que se componen los documentos JSON: arrays, objetos, cadenas de texto, números y booleanos.

Además hay que tener en cuenta las interfaces que implementan estos objetos, ya que nos ofrecen una serie de métodos que nos permiten navegar a través de un documento JSON.

- `JsonObject` → `Map<String,JsonValue>` que, entre otros, nos ofrece métodos como:
 - `isEmpty`: indica si el objeto JSON está vacío.
 - `size`: devuelve el número de pares clave:valor del objeto.
 - `containsKey`: indica si existe la clave entre los pares clave:valor del objeto.
 - `keySet`: devuelve un conjunto con las claves, partes izquierdas, presentes en el objeto.
 - `values`: devuelve una colección `Collection<JsonValue>` con los valores del objeto.
 - Cualquier operación de modificación del objeto JSON como puede ser `put`, `remove`, ... producirá una `UnsupportedOperationException` ya que un `JsonObject` es inmutable.
- `JsonArray` → `Collection<JsonValue>`, `Iterable<JsonValue>`, `List<JsonValue>` que entre otros, nos ofrece métodos como:
 - `isEmpty`: indica si el array JSON está vacío.
 - `size`: devuelve el número de elementos del array JSON.
 - `subList`: nos permite obtener, como lista, una sección del array JSON.
 - Cualquier operación de modificación del array JSON como puede ser `add`, `remove`, ... producirá una `UnsupportedOperationException` ya que un `JsonArray` es inmutable.
 - Un `JsonArray` es iterable.

<div>COLEXIO</div> <div>VIVAS S.L.</div>	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

Usando los métodos anteriores vamos a ver unos ejemplos de navegación por un árbol JSON anterior.

Ejemplo 4

```
// Creamos una elemento Json mediante código usando
// el ejemplo 4 del punto 3.4 o usando el fichero peliculas.json
JSONArray peliculas=json.creaArray();

// Si cada elemento del array inicial es una película, ¿cuántas películas hay?
System.out.println("Nº películas: "+peliculas.size());

// ¿Cual es el título de cada película?
for (JsonValue peli : peliculas) { // peli es de tipo JsonObject
    System.out.println("Titulo: "+ peli.asJsonObject().getString("titulo"));
    System.out.println("Titulo: "+((JsonObject)peli).getString("titulo"));
}

System.out.println("");
for (JsonObject peli : peliculas.getValuesAs(JsonObject.class)){
    System.out.println("Titulo: "+peli.getString("titulo"));
}

System.out.println("");
for (int i = 0; i < peliculas.size(); i++) {
    System.out.println("Titulo: "+peliculas.getJSONObject(i).getString("titulo"));
}

// ¿De qué año es cada película?
System.out.println("");
for (JsonObject peli : peliculas.getValuesAs(JsonObject.class)){
    System.out.println("Año: "+peli.getInt("año"));
}

// Entre los atributos de las películas. ¿Existe el campo sinopsis?
for (JsonObject peli : peliculas.getValuesAs(JsonObject.class)) {
    System.out.println("Sinopsis: "+peli.containsKey("sinopsis"));
}

// ¿Que atributos tiene la primera película?
Set<String> atributos=peliculas.getJSONObject(0).keySet();
for (String atri:atributos) {
    System.out.println(atri);
}

// ¿Es alguna película del 2010?
for (JsonValue peli : peliculas) {
    System.out.println("Es del 2010: "+
        (((JsonObject)peli).getInt("año")==2010?"Si":"No"));
}
```

<div>COLEXIO</div> <div>VIVAS</div> <div>S.L.</div>	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

```
// ¿Que valores tiene la segunda película?
Collection<JsonValue> colec=peliculas.getJSONObject(1).values();
for (JsonValue valor:colec) {
    System.out.println("Valor: "+valor.toString());
}

// Para cada película, obtén los nombres de los interpretes
for (JsonObject peli : peliculas.getValuesAs(JsonObject.class)){
    System.out.println("\nTitulo: "+peli.getString("titulo"));
    JSONArray interpretes=peli.getJSONArray("interpretes");
    for (JsonObject interprete : interpretes.getValuesAs(JsonObject.class)){
        System.out.println("Nombre: "+interprete.getString("nombre"));
    }
}
```

3.4. Creación de un modelo de objetos de código de aplicación

Un árbol JSON es creado mediante las llamadas anidadas a los métodos add pertenecientes a las clases JsonObjectBuilder y JsonArrayBuilder y construido mediante una llamada al método build.

La interface JsonBuilderFactory nos ofrece los métodos createArrayBuilder y createObjectBuilder los cuales retornan un objeto JsonArrayBuilder y JsonObjectBuilder respectivamente.

JsonObjectBuilder y JsonArrayBuilder poseen métodos add similares pero mientras los métodos de JsonObjectBuilder tienen siempre dos atributos (clave y valor) los métodos add de JsonArrayBuilder solo tienen un atributo (el valor). Ambos poseen el método addnull para añadir un nulo a un objeto o array JSON ya que no se permite usar un nulo en los métodos add.

Se pueden anidar matrices y objetos pasando un nuevo objeto JsonArrayBuilder o un nuevo objeto JsonObjectBuilder al método add correspondiente.

<div>COLEXIO</div> <div>VIVAS S.L.</div>	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

Imaginemos que queremos construir el siguiente árbol JSON:

```
[
  {
    "titulo": "El atlas de las nubes",
    "año": 2012,
    "directores": "Lana Wachowski, Lilly Wachowski",
    "interpretes": [
      {
        "nombre": "Tom Hanks",
        "fechaNacimiento": {
          "año": 1956,
          "mes": 8
        }
      },
      {
        "nombre": "Halle Berry",
        "fechaNacimiento": {
          "año": 1966,
          "mes": 7
        }
      }
    ]
  },
  {
    "titulo": "La red social",
    "año": 2010,
    "directores": "David Fincher",
    "interpretes": [
      {
        "nombre": "Jesse Eisenberg",
        "fechaNacimiento": {
          "año": 1983,
          "mes": 9
        }
      },
      {
        "nombre": "Andrew Garfield",
        "fechaNacimiento": {
          "año": 1983,
          "mes": 7
        }
      }
    ]
  }
]
```

<div>COLEXIO</div> <div>VIVAS</div> <div>S.L.</div>	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

El código para crear el árbol JSON anterior es el siguiente:

Ejemplo 5

```

public JSONArray creaArray() {
    JSONArray array= (JSONArray) Json.createArrayBuilder()
        .add(Json.createObjectBuilder()
            .add("titulo", "El atlas de las nubes")
            .add("año", 2012)
            .add("directores", "Lana Wachowski, Tom Tykwer, Lilly Wachowski")
            .add("interpretes", Json.createArrayBuilder()
                .add(Json.createObjectBuilder()
                    .add("nombre", "Tom Hanks")
                    .add("fechaNacimiento", Json.createObjectBuilder()
                        .add("año", "1956")
                        .add("mes", 8)
                    )
                )
                .add(Json.createObjectBuilder()
                    .add("nombre", "Halle Berry")
                    .add("fechaNacimiento", Json.createObjectBuilder()
                        .add("año", "1966")
                        .add("mes", 7)
                    )
                )
            )
        )
        .add(Json.createObjectBuilder()
            .add("titulo", "La red social")
            .add("año", 2010)
            .add("directores", "David Fincher")
            .add("interpretes", Json.createArrayBuilder()
                .add(Json.createObjectBuilder()
                    .add("nombre", "Jesse Eisenberg")
                    .add("fechaNacimiento", Json.createObjectBuilder()
                        .add("año", "1983")
                        .add("mes", 9)
                    )
                )
                .add(Json.createObjectBuilder()
                    .add("nombre", "Andrew Garfield")
                    .add("fechaNacimiento", Json.createObjectBuilder()
                        .add("año", "1983")
                        .add("mes", 7)
                    )
                )
            )
        )
        .build();
    return array;
}

```

<div>COLEXIO</div> <div>VIVAS</div> <div>S.L.</div>	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					


Además de crear el árbol JSON en memoria podemos, mediante el modelo en Streaming, crearlo directamente en disco mediante interfaz JsonGenerator, la cual funciona de forma parecida al ejemplo anterior, pero en vez de usar métodos add usaremos métodos write.

El siguiente ejemplo veremos cómo escribir directamente a disco la primera parte del ejemplo anterior:

Ejemplo 6

```
public void generaEndisco(File f) throws FileNotFoundException{
    JsonGenerator generator = Json.createGenerator(new
                                                FileOutputStream(f));

    generator.writeStartArray()
        .writeStartObject()
            .write("titulo", "El atlas de las nubes")
            .write("año", 2012)
            .write("directores", "Lana Wachowski, Lilly Wachowski")
            .writeStartArray("intepretes")
                .writeStartObject()
                    .write("nombre", "Tom Hanks")
                    .writeStartObject("fechaNacimiento")
                        .write("año", "1956")
                        .write("mes", 8)
                    .writeEnd()
                .writeEnd()
            .writeStartObject()
                .write("nombre", "Halle Berry")
                .writeStartObject("fechaNacimiento")
                    .write("año", "1966")
                    .write("mes", 7)
                .writeEnd()
            .writeEnd()
        .writeEnd()
    .writeEnd()
    .close();
}
```

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

4. APIs de servicios web

Veremos ahora unas cuantas listas con servicios web que se pueden usar:

- Buscador de datos ofrecidos por el Gobierno de España.
https://datos.gob.es/es/catalogo?res_format_label=JSON
- Buscador de datos ofrecidos por la Xunta de Galicia.
<https://abertos.xunta.gal/busca-de-datos>
- Servicios ofrecidos por MeteoGalicia.
https://www.meteogalicia.gal/web/RSS/rssIndex.action?request_locale=gl
- Listado de APIs públicas para usar en el desarrollo de software.
<https://github.com/toddmotto/public-apis>
- Buscador de APIs.
<https://www.programmableweb.com/category/all/apis?&order=created&sort=desc>

De las siguientes veremos una pequeña descripción de su utilización:

4.1. Open Weather Map

Openweathermap nos ofrece una completa API <http://openweathermap.org/api> para poder obtener el pronóstico del tiempo en una o varias localidades.

Por defecto nos ofrece la información en formato JSON pero mediante el parámetro `&mode=xml` podemos obtener el XML equivalente.


Otro parámetro interesante es `&units=metric` que convierte todas las temperaturas a grados centígrados.

En todas nuestras consultas usaremos a API KEY generada para pruebas de la siguiente forma:

`&APPID=a975f935caf274ab016f4308ffa23453`

Entre las opciones que disponemos se encuentran:

- Listar el tiempo de una localidad por su nombre, se le puede añadir (es opcional) el código de país para determinar el país de la localidad:
<http://api.openweathermap.org/data/2.5/weather?q=vigo,es&lang=es&APPID=a975f935caf274ab016f4308ffa23453>
- Listar el tiempo de una localidad por su id:
<http://api.openweathermap.org/data/2.5/weather?id=3105976&lang=es&APPID=a975f935caf274ab016f4308ffa23453>
- Listar el tiempo de varias ciudades por sus códigos:
<http://api.openweathermap.org/data/2.5/group?id=3105976,3119841,3113157&lang=es&APPID=a975f935caf274ab016f4308ffa23453>
- Utilizando su longitud y latitud:
<http://api.openweathermap.org/data/2.5/weather?lat=42.232819&lon=-8.72264&APPID=a975f935caf274ab016f4308ffa23453>

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

- Si queremos encontrar las N predicciones mas próximas a una un punto podemos usar:

<http://api.openweathermap.org/data/2.5/find?lat=42.232819&lon=-8.72264&cnt=20&APPID=a975f935caf274ab016f4308ffa23453>

La descripción de cada campo devuelto se puede consultar en:

<http://openweathermap.org/weather-data>

Podemos obtener un listado de poblaciones disponibles en:

<http://78.46.48.103/sample/city.list.json.gz>

Los iconos utilizados se puede descargar cambiando el nombre de la imagen (el nombre se obtiene en las peticiones de datos) en:

<http://openweathermap.org/img/w/10d.png>.

4.2. Evenful


[Evenful](#) nos proporciona unos servicios web similares a los proporcionados por Last.Fm, pero además nos permite buscar eventos cercanos a una localidad. Su API está disponible en: <http://api.eventful.com/>

Para realizar búsquedas deberemos disponer de una key válida. Para pruebas podemos usar la API KEY:

&app_key=c2tPtVFTrSk8xnQS

Ejemplos de búsquedas son:

- Eventos musicales cercanos a A Coruña (se usan + para concatenar tópicos de búsqueda):
http://api.eventful.com/json/events/search?q=music&l=a+coruna&app_key=c2tPtVFTrSk8xnQS
- Eventos en Vigo:
http://api.eventful.com/json/events/search?l=vigo&app_key=c2tPtVFTrSk8xnQS
- Restaurantes en España:
http://api.eventful.com/json/venues/search?keywords=Restaurant&location=spain&app_key=c2tPtVFTrSk8xnQS
- Listar eventos musicales en Vigo, limitando a 25 resultados por página, las distancias en kilometros y busque e un radio de 250 kilometros (la unidad es la del parámetro units):
http://api.eventful.com/json/events/search?q=music&l=vigo&units=km&within=250&page_size=25&app_key=c2tPtVFTrSk8xnQS
- Información de un local por id: Estadio de Balaidos:
http://api.eventful.com/json/venues/get?app_key=c2tPtVFTrSk8xnQS&id=VO-001-010832069-6
- Información de un local por id: Un partido del celta:
http://api.eventful.com/json/events/get?app_key=c2tPtVFTrSk8xnQS&id=EO-001-098508729-3

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

4.3. Last.fm

LastFM ofrece una API <http://www.lastfm.es/api/rest> que proporciona todo tipo de información sobre artistas, grupos musicales, álbumes, canciones e incluso geolocalización de actividades cercanas a una longitud y latitud.

Nos ofrece información tanto en formato XML como en formato JSON; por defecto los resultados están en formato XML, para recuperar los resultados en formato JSON hay que añadir el parámetro: **&format=json**.

Para realizar consultas en LastFM se requiere estar registrado y contar con una API KEY válida. Podemos usar la API KEY **&api_key=7ba9be9fd8a0a0c23c81c9e368c7e120** procedente de una cuenta registrada expreso para pruebas.


Entre otras, nos ofrece las siguientes búsquedas:

- Buscar por álbum:
http://ws.audioscrobbler.com/2.0/?method=album.search&album=believe&api_key=7ba9be9fd8a0a0c23c81c9e368c7e120&format=json
- Información de un álbum:
http://ws.audioscrobbler.com/2.0/?method=album.getinfo&artist=Cher&album=Believe&api_key=7ba9be9fd8a0a0c23c81c9e368c7e120&format=json
- Información de un artista:
http://ws.audioscrobbler.com/2.0/?method=artist.getinfo&artist=Cher&api_key=7ba9be9fd8a0a0c23c81c9e368c7e120&format=json
- Buscar por artista:
http://ws.audioscrobbler.com/2.0/?method=artist.search&artist=cher&api_key=7ba9be9fd8a0a0c23c81c9e368c7e120&format=json

4.4. Wikipedia

En la siguiente página https://www.mediawiki.org/wiki/API:Main_page/es podemos ver la API de Wikipedia para realizar consultas contra ella. Podemos consultar los parametros necesarios para hacer búsquedas en <http://commons.wikimedia.org/w/api.php>.

- Búsqueda de tópicos
<http://commons.wikimedia.org/w/api.php?action=opensearch&search=vigo&limit=50>
- Acceso a un tópico
<http://en.wikipedia.org/w/api.php?format=json&action=query&titles=vigo&prop=revisions&rvprop=content>

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

4.5. IMDB


Mediante la web <http://www.omdbapi.com> se puede buscar en la base de datos de IMDB (www.imdb.com) usando los siguientes criterios de búsqueda:

Parámetro	Descripción
s	String - Búsqueda por título de la película
y	String - Filtra una búsqueda por año
i	String - Devuelve película por su id IMDB
t	String - Devuelve película por su título
r	JSOM / XML - Tipo de datos a devolver JSOM o XML
plot	Short / full - Argumento de la película.
tomatoes	True - Añade información de: http://www.rottentomatoes.com

Y usando la siguiente API KEY generada para pruebas: **&apikey=75fe1286**

Ejemplos de búsquedas son:

- Búsqueda por películas que en el título contenga la cadena eleven:
<http://www.omdbapi.com/?s=eleven&apikey=75fe1286>
- Búsqueda la película con título Ocean's Eleven en el año 2001:
<http://www.omdbapi.com/?t=Ocean%27s%20Eleven&y=2001&apikey=75fe1286>
- Búsqueda a película con título Ocean's Eleven del año 2001 y devuelve su argumento (plot) junto con los datos de rotten tomatoes:
[http://www.omdbapi.com/?t=Ocean's Eleven&y=2001&plot=full&tomatoes=true&apikey=75fe1286](http://www.omdbapi.com/?t=Ocean's%20Eleven&y=2001&plot=full&tomatoes=true&apikey=75fe1286)

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD COMPETENCIA:							

5. Apéndice 1: API de Google


Google ofrece sus APIs para realizar consultas en formatos JSOM y XML. En las últimas versiones Google pide tener una API para poder hacer uso de sus servicios. Por este motivo no la usaremos.

Disponemos de las siguientes APIs:

- Rutas: <https://developers.google.com/maps/documentation/directions/?hl=es>
- Distancias: <https://developers.google.com/maps/documentation/distance-matrix/start>
- Elevaciones de terreno: <https://developers.google.com/maps/documentation/elevation/start>
- Geolocalización: <https://developers.google.com/maps/documentation/geocoding/start>
- Lugares: <https://developers.google.com/places/web-service/intro>
- Carreteras: <https://developers.google.com/maps/documentation/roads/intro>
- Zonas horarias: <https://developers.google.com/maps/documentation/timezone/start>

Ejemplo de consultas:

- Búsquedas sobre calles: En el ejemplo siguiente buscamos la calle Príncipe situada en Vigo.
<https://maps.googleapis.com/maps/api/geocode/json?address=calle principe, Vigo, España>
 En donde la dirección de viene en formato: Calle, Población, Provincia / Estado, País
- Ruta entre Vigo y Ourense:
<https://maps.googleapis.com/maps/api/directions/json?origin=Vigo&destination=ourense>

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

6. Apéndice 2: Evitar bloqueo de peticiones


El siguiente código permite superar los bloqueos que ciertos servicios web ponen a los Bots.

El User-Agent se utiliza para simular que la petición la hace un navegador web.

Ejemplo

```
public JsonValue leerHttpsReferer(String direccion) throws IOException {
    URL url = new URL(direccion);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestProperty("User-Agent", "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.4; en-US; rv:1.9.2.2) Gecko/20100316 Firefox/3.6.2");

    try (InputStreamReader isr =
        new InputStreamReader(conn.getInputStream(), Charset.forName("UTF-8"));
        BufferedReader breader = new BufferedReader(isr);
        JsonReader reader = Json.createReader(breader);
    ) {
        return reader.read();
    } finally {
        conn.disconnect();
    }
}
```

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

7. Bibliografía

1. Herramientas

1. [jsonviewer](#)
2. [jsonformatter](#)
3. [jsonlint](#)

2. Json

1. [Tutorial del modelo de Objetos de JSON](#)
2. [Página web oficial](#)
3. [APi Json](#)
4. [APi Json](#)

3. Descarga JAR y JavaDoc

1. [Descarga de los Jar de Json y con el Javadoc](#)
2. [Descarga alternativo de los Jar de Json y con el Javadoc](#)

4. Componentes

1. JsonValue

1. <https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonValue.html>
2. <https://docs.oracle.com/javaee/7/api/javax/json/JsonValue.html>

2. JsonStructure

1. <https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonStructure.html>
2. <https://docs.oracle.com/javaee/7/api/javax/json/JsonStructure.html>

3. JsonObject

1. <https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonObject.html>
2. <https://docs.oracle.com/javaee/7/api/javax/json/JsonObject.html>

4. JsonArray

1. <https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonArray.html>
2. <https://docs.oracle.com/javaee/7/api/javax/json/JsonArray.html>

5. JsonException


1. <https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonException.html>
2. <https://docs.oracle.com/javaee/7/api/javax/json/JsonException.html>

6. JsonString

1. <https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonString.html>
2. <https://docs.oracle.com/javaee/7/api/javax/json/JsonString.html>

7. JsonNumber

1. <https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonNumber.html>
2. <https://docs.oracle.com/javaee/7/api/javax/json/JsonNumber.html>

	RAMA:	Informática	CICLO:	Desenvolvemento de Aplicacions Multiplataforma				
	MÓDULO:	Acceso a datos					CURSO:	2º
	PROTOCOLO:	Apuntes clases		AVAL:	1	DATA:	2022/2023	
	UNIDAD		COMPETENCIA:					

8. JsonGenerator

1. <https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/stream/JsonGenerator.html>
2. <https://docs.oracle.com/javaee/7/api/javax/json/stream/JsonGenerator.html>

9. JsonReader

1. <https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonReader.html>
2. <https://docs.oracle.com/javaee/7/api/javax/json/JsonReader.html>

10. JsonWriter

1. <https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonWriter.html>
2. <https://docs.oracle.com/javaee/7/api/javax/json/JsonWriter.html>

11. JsonObjectBuilder

1. <https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonObjectBuilder.html>
2. <https://docs.oracle.com/javaee/7/api/javax/json/JsonObjectBuilder.html>

12. JsonArrayBuilder

1. <https://javadoc.io/static/javax.json/javax.json-api/1.1.4/javax/json/JsonArrayBuilder.html>
2. <https://docs.oracle.com/javaee/7/api/javax/json/JsonArrayBuilder.html>

5. Apis de servicios web

1. [Buscador de datos ofrecidos por el Gobierno Español.](#)
2. [Servicios ofrecidos por MeteoGalicia.](#)
3. [Listado de APIs públicas para usar en el desarrollo de software.](#)
4. [Buscador de Apis](#)