

**UNIVERSIDAD CIENTÍFICA DEL SUR**

**FACULTAD DE CIENCIAS EMPRESARIALES**

**CARRERA DE INGENIERÍA DE SISTEMAS DE INFORMACIÓN Y GESTIÓN**

**Asignatura :** Lenguaje de Programación

**Profesor :** Eric Gustavo Coronel Castillo

**Tema :** API JSON de Java

**Alumnos :**

Becerra Gutiérrez, Walter

Zavala Alvarez, Carlos

Lima, 22 de Febrero Del 2017

INDICE

1. Indice 1
2. Resume 3
3. Antecedentes 4
4. Definiciones a utilizar 5
5. Fundamentos Teóricos 8
6. Casos demostrativos 17
7. Discusión 20
8. Conclusiones 21
9. Recomendaciones 21
10. Bibliografia 22

**2. Resumen**

En los últimos años El internet y las APIs (Interfaces de [programación](http://laguiatv.abc.es/programacion/) de aplicaciones ) van evolucionando y facilitan el intercambio de información de manera flexible y ágil gracias a los múltiples servicios disponibles en la red. Es importante considerar las tecnologías empleadas en la implementación de los servicios de Internet. como son los servicios Web, los cuales usan comúnmente 2 tipos de protocolos:

SOAP (Simple Object Access Protocol) Protocolo estándar de acceso, utilizada en XML

REST (Representational State Transfer)Transferencia de Estado Representacional. JSON

La diferencia básica entre estos dos protocolos es en cuanto al tipo de mensajes que se intercambian; para el caso de SOAP, los mensajes son en formato WSDL Web Services Description Language (basado en XML); mientras que en el caso de REST, el tipo de mensaje más difundido es JSON, el cual basa su sintaxis en el lenguaje JavaScript. Los mensajes WSDL al estar basados en el lenguaje XML son complejos comparados con los de tipo JSON, razón por la cual las redes sociales más difundidas com Facebook, Twitter han sido implementados y tienen interfaces abiertas para su desarrollo con el protocolo REST-JSON.

JSON utiliza su propia librería, antiguamente usaba Gson.jar de Google, SimpleJson.jar

Para validar JSON utiliza La libreria mongoose que ES libreria de JavaScrip.

**3. Antecedentes**

Antes de que apareciera JSON, se utilizaba:

1. XML eXtensible Markup Language (lenguaje de marcado extensible) Se utilizaba para la comunicación de datos vía remota entre componentes ó para la comunicación de datos entre aplicaciones.

Ventajas de XML

Seguridad en el intercambio de los datos de gobierno a gobierno ó entre empresas como ejemplo los bancos.

Desventaja de XML

Al usar etiquetas para la descripción y organizacion de datos, el tamaño del documento aumenta.

Para realizar la transferencia vía remota, se necesita transferir en menor tamaño, en muchas aplicaciones JSON reemplaza al XML por ser mas liviano

1. Al ser un formato de JavaScrip, es mas fácil trabajar datos en formato JSON que trabajarlos em formato XML con el browse como inicialmente se hacía
2. Los servicios Web basados en XML, se han visto reemplazados por formatos JSON a los cuales se les llama servicios REST o RESTfull.
3. Antes de que apareciera el API JSON de Java, se tenía que utilizar las librerias de terceros como:

Gson.jar de Google, SimpleJson.jar

Ahora Java tiene su propia libreria y ya no es necesario utilizarlas

**4.Definiciones a utilizar**

API (Application Programming Interfaces )

(Interfaces de [programación](http://laguiatv.abc.es/programacion/) de aplicaciones) una API es una especificación formal sobre cómo un módulo de un software se comunica o interactúa con otro.

En otras palabras, las API son un conjunto de comandos, funciones y protocolos informáticos que permiten a los desarrolladores crear programas específicos para ciertos sistemas operativos. Las API simplifican en gran medida el trabajo de un creador de programas, ya que no tiene que escribir códigos desde cero. Estas permiten al informático usar funciones predefinidas para interactuar con el sistema operativo o con otro programa.

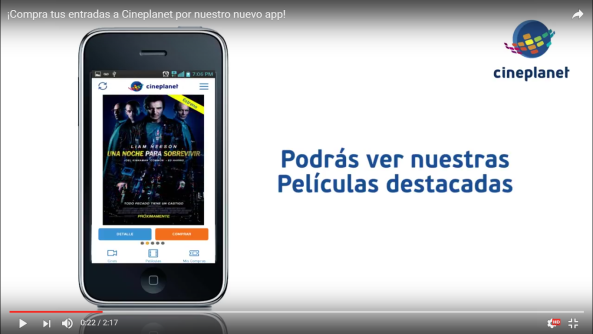
La API es una interfaz o rostro que sólo da la cara al software. Es decir, el usuario no ve eso. Con las API, las aplicaciones como Facebook, Twitter, y Candy Crush se pueden comunicar entre ellas sin que el usuario tenga que intervenir.

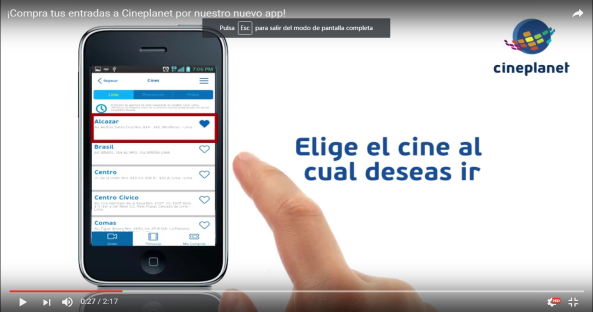
Por ejemplo, cuando el usuario compra entradas a través de la página web de una sala de [cine](http://hoycinema.abc.es/) e introduce la información de su tarjeta de crédito, la web usa una API para enviar dicha información de forma remota a otro programa que verifica si los datos bancarios son correctos. Una vez que se confirma el pago, la aplicación remota envía la información al sitio web del [cine](http://hoycinema.abc.es/) y le da un «OK», por lo que esta página emite los tickets.

En todo ese proceso, el usuario solo ve una cara del proceso, la página del cine, pero detrás hay muchas aplicaciones que se están comunicando gracias a las API.

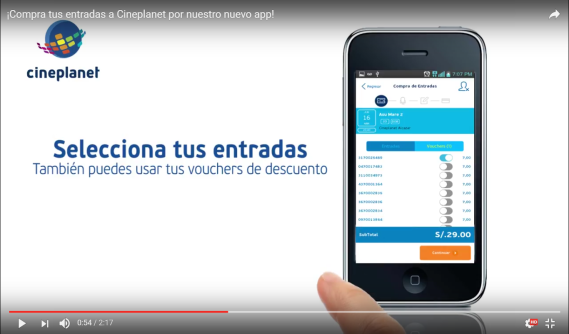
Los usuarios pueden ver muchas API en todos lados aunque no se percaten de ello. Por ejemplo, en la página de la universidad científica hay una columna Social que tiene iconos de redes sociales, como Facebook y Twitter. Estos son enlaces que se encargan de llamar a las APIs asociadas a esos servicios para que el usuario pueda tuitear o compartir la información en Facebook sin tener que salir de la página web.

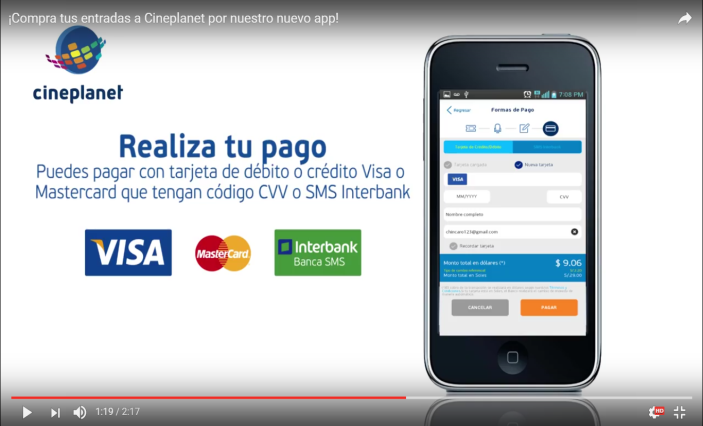
**Ejemplo para compra entradas al cine através de aplicativos**





**Ejemplo para compra entradas al cine através de aplicativos**





Las Interface de programación de aplicaciones, se tratan de un conjunto de librerias de código JAVA compilado que se incluyen en la Java Runtime Environment (JRE) o Entorno de Ejecución JAVA.  
Algunos paquetes de la API JAVA son:  
  
- java.lang: contiene clases esenciales como números, strings, objetos, seguridad y threads. Es el único paquete que se incluye automáticamente en todo programa Java.  
  
- java.io: contiene las clases que manejan la Entrada/Salida, Serialización de objetos.  
  
- java.util: contiene clases útiles que permiten manejar estructuras de datos o colecciones, fechas, hora, excepciones, etc.  
  
- java.net: contiene clases como URL, TCP, UDP, IP, etc. que permiten implementar aplicaciones distribuidas. Provee soporte para sockets.  
  
- java.awt: contiene clases para el manejo de la GUI, pintar gráficos e imágenes.  
  
- java.applet: contiene clases útiles para la creación y manipulación de applets y recursos para reproducción de audio.  
  
- java.sql: contiene clases para el manejo de base de datos relaciones.

**Servicios SOAP** (Simple Object Access Protocol)

Es un [protocolo](https://es.wikipedia.org/wiki/Protocolo_(inform%C3%A1tica)) [estándar](https://es.wikipedia.org/wiki/Norma_(tecnolog%C3%ADa)) que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos [XML](https://es.wikipedia.org/wiki/XML). Este protocolo deriva de un protocolo creado por [Dave Winer](https://es.wikipedia.org/w/index.php?title=Dave_Winer&action=edit&redlink=1" \o "Dave Winer (aún no redactado)) en 1998, llamado [XML-RPC](https://es.wikipedia.org/wiki/XML-RPC). SOAP fue creado por [Microsoft](https://es.wikipedia.org/wiki/Microsoft), [IBM](https://es.wikipedia.org/wiki/IBM) y otros. Está actualmente bajo el auspicio de la [W3C](https://es.wikipedia.org/wiki/World_Wide_Web_Consortium). Es uno de los protocolos utilizados en los [servicios Web](https://es.wikipedia.org/wiki/Servicios_Web).

**Servicios REST** (Representational State Transfer) Transferencia de Estado Representacional. Este servicio plantea un estilo de arquitectura cliente-servidor en la cual un servicio es visto como un recurso es identificado a través de una dirección URL, mediante la cual puede ser accedido o consumido. Para acceder a estos servicios web, se hace uso de mensajes en formato simple, los cuales se intercambian entre cliente y servidor. REST define a partir de HTTP, cuatro métodos:

GET, PUT, DELETE y POST, de los cuales los más usados son: GET y PUT.

GET: Este método es usado para enviar la representación de un recurso o servicio al cliente. PUT: Este método es usado para transferir el estado de un cliente al recurso.

Para la comunicación e intercambio de información entre cliente y servidor a través de REST, se puede hacer uso de diversos formatos y lenguajes: XML, HTML, JSON.

Debido a la sencillez de los mensajes JSON, este es el tipo de mensajes mas difundido en diferentes servicios propios de redes sociales (Facebook y Twitter) y comunidades en Internet.

Los sistemas que siguen los principios REST se llaman con frecuencia RESTful.

**5. Fundamentos Teóricos**

[**JSON**](http://json.org/) (JavaScript Object Notation) es un formato para intercambio de datos liviano, basado en texto e independiente del lenguaje de programación, que resulta fácil de escribir y leer tanto para los seres humanos como para las máquinas.

JSON puede representar dos tipos estructurados: objetos y matrices.

1.**Un objeto,** es una colección no ordenada de cero o más pares de nombres/valores. Esto se realiza como un objeto, registro, estructura, diccionario, tabla hash, lista con clave o matriz asociativa.

2.**Una matriz,** es una secuencia ordenada de cero o más valores. Los valores pueden ser cadenas, números, booleanos, nulos y estos dos tipos estructurados.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

En JSON, se presentan de estas formas:

**Un *objeto***, es un conjunto desordenado de pares nombre/valor. Un objeto comienza con **{** (llave de apertura) y termine con **}** (llave de cierre). Cada nombre es seguido por **:** (dos puntos) y los pares nombre/valor están separados por **,** (coma).



{

    "firstName": "John",

    "lastName": "Smith"

}

**Un *arreglo***, es una colección de valores. Un arreglo comienza con **[** (corchete izquierdo) y termina con **]** (corchete derecho). Los valores se separan por **,** (coma).



"phoneNumbers": [

        {

            "type": "home",

            "number": "212 555-1234"

        },

        {

            "type": "fax",

            "number": "646 555-4567"

        }

    ]

**Un *valor***,  puede ser una *cadena de caracteres* con comillas dobles, o un *número*, o **true** o **false** o **null**, o un *objeto* o un *arreglo*. Estas estructuras pueden anidarse.



"address": {

        "streetAddress": "21 2nd Street",

        "city": "New York",

        "state": "NY",

        "postalCode": 10021,

"resident": false

    },

**Una *cadena de caracteres***, es una colección de cero o más caracteres Unicode, encerrados entre comillas dobles, usando barras divisorias invertidas como escape. Un carácter está representado por una cadena de caracteres de un único carácter. Una *cadena de carateres* es parecida a una cadena de caracteres C o Java.



"experiencia": {

        "como profesor de LP, tiene una experiencia,

de 10 años "

    },

**Un *número***, es similar a un número C o Java, excepto que no se usan los formatos octales y hexadecimales.



"phoneNumbers": [

        {

            "type": "home",

            "number": 4712826,

"type": "celular",

            "number": 994747223

        },

]

Como ejemplo en el Listado 1,se muestra la representación JSON de un objeto que describe a una persona. El objeto tiene valores de cadena para nombre y apellido, un valor numérico para la edad, un valor de objeto que representa el domicilio de la persona y un valor de matriz de objetos de números telefónicos.

Listado 1. Ejemplo de representación de un objeto en JSON

{

    "firstName": "John",

    "lastName": "Smith",

    "age": 25,

    "address": {

        "streetAddress": "21 2nd Street",

        "city": "New York",

        "state": "NY",

        "postalCode": 10021

    },

    "phoneNumbers": [

        {

            "type": "home",

            "number": "212 555-1234"

        },

        {

            "type": "fax",

            "number": "646 555-4567"

        }

    ]

}

JSON suele utilizarse en aplicaciones Ajax, configuraciones, bases de datos y servicios web RESTful. Todos los sitios web populares ofrecen JSON como formato para intercambio de datos con sus servicios web RESTful.

**Procesamiento JSON**   
La API de Java para procesamiento JSON ([JSR 353](http://jcp.org/en/jsr/detail?id=353)) proporciona rutinas API portátiles que permiten analizar, generar, transformar y consultar JSON usando rutinas API de modelos de objetos y de streaming.

**La API de modelos de objetos:** Crea una estructura de árbol, de acceso aleatorio, que representa los datos JSON almacenados en la memoria. Es posible recorrer el árbol y formular consultas. Este modelo de programación es el más flexible y posibilita el procesamiento en casos en que se requiera acceso aleatorio a la totalidad del contenido de la memoria. Sin embargo, a menudo no es tan eficiente como el modelo de streaming y requiere más memoria.

**La API de streaming:**Ofrece un modo de analizar y generar JSON en streams. Le otorga al programador el control sobre el análisis y la generación. La API de streaming ofrece un analizador basado en eventos y brinda al desarrollador de aplicaciones la posibilidad de pedir el evento siguiente en lugar de tener que ocuparse del evento en una devolución de llamada. De este modo, el desarrollador cuenta con mayor control procedimental del procesamiento JSON. El código de aplicación puede procesar o descartar el evento del analizador y pedir el siguiente evento (extraer el evento). El modelo de streaming es adecuado para el procesamiento local cuando no se requiere acceso aleatorio a otras porciones de la información. De manera similar, la API de streaming permite generar JSON bien formado en stream escribiendo un evento por vez.

La API de modelos de objetos   
La API de modelos de objetos es similar a la API de modelos de objetos de documento (DOM) para XML. Es una API de alto nivel que proporciona modelos de objetos inmutables para estructuras de objetos y matrices JSON. Estas estructuras JSON se representan como modelos de objetos usando los tipos de Java [JsonObject](http://docs.oracle.com/javaee/7/api/javax/json/JsonObject.html) y [JsonArray](http://docs.oracle.com/javaee/7/api/javax/json/JsonArray.html). En la Tabla 1 se incluyen las clases e interfaces principales de la API de modelos de objetos.

**JsonObject** suministra una vista Map para obtener acceso a la colección no ordenada de cero o más pares de nombres/valores del modelo. De modo similar,

**JsonArray** ofrece una vista List para obtener acceso a la secuencia ordenada de cero o más valores del modelo.

Tabla 1. Clases principales de la API de modelos de objetos

|  |  |
| --- | --- |
| Clase o interfaz | Descripción |
| Json | Contiene métodos estáticos para crear lectores, escritores, constructores de JSON y sus objetos de fábrica. |
| JsonGenerator | Escribe datos JSON en forma de stream, con un valor por vez. |
| JsonReader | Lee datos JSON de un stream y crea un modelo de objeto en la memoria. |
| JsonObjectBuilder JsonArrayBuilder | Crean un modelo de objeto o un modelo de matriz en la memoria agregando valores del código de aplicación. |
| JsonWriter | Escribe un modelo de objeto de la memoria en un stream. |
| JsonValue JsonObject JsonArray JsonString JsonNumber | Representan tipos de datos para valores en datos JSON. |

JsonObject, JsonArray, JsonString y JsonNumber son subtipos de JsonValue. Hay constantes definidas en la API para valores JSON nulos, verdaderos y falsos.   
La API de modelos de objetos usa patrones generadores para crear estos modelos de objetos desde cero. El código de aplicación puede usar la interfaz JsonObjectBuilder para crear modelos que representen objetos JSON. El modelo que se obtiene es del tipo JsonObject. El código de aplicación puede usar la interfaz JsonArrayBuilder para crear modelos que representen matrices JSON. El modelo que se obtiene es del tipo JsonArray.   
Estos modelos de objetos también pueden crearse a partir de un origen de entrada (por ejemplo InputStream o Reader) usando la interfaz JsonReader. De modo similar, los modelos de objetos pueden escribirse en un origen de salida (por ejemplo OutputStream o Writer) usando la clase JsonWriter.   
Por ejemplo, escribamos código para efectuar una búsqueda en los posts públicos de Facebook usando la API de modelos de objetos. La API de Facebook devuelve los resultados de la búsqueda en el formato JSON que se muestra en el Listado 2.

{

     "data" : [

         { "from" : { "name" : "xxx", ... }, "message" : "yyy", ... },

         { "from" : { "name" : "ppp", ... }, "message" : "qqq", ... },

         ...

     ], ...

}

Listado 2. Representación JSON de una búsqueda en los posts públicos de Facebook   
  
Podemos usar la API de modelos de objetos para obtener nombres y sus posts públicos relacionados con el término java. En el Listado 3, las líneas 1 a 3 crean JsonReader; la línea 5 crea JsonObject para los resultados; la línea 7 itera respecto de cada resultado; y las líneas 8 a 11 obtienen el nombre de la persona que publicó el post y el post en sí, y los imprimen. Nótese que JsonReader y otros objetos de esta API pueden usarse en la instrucción try-with-resources (que también se denomina gestión automática de recursos [ARM]).

URL url = new URL("https://graph.facebook.com/search?q=java&type=post");

try (InputStream is = url.openStream();

     JsonReader rdr = Json.createReader(is)) {

     JsonObject obj = rdr.readObject();

     JsonArray results = obj.getJsonArray("data");

     for (JsonObject result : results.getValuesAs(JsonObject.class)) {

         System.out.print(result.getJsonObject("from").getString("name"));

         System.out.print(": ");

         System.out.println(result.getString("message", ""));

         System.out.println("-----------");

     }

}

Listado 3. Procesamiento de posts de Facebook con la API de modelos de objetos

La API de streaming   
La API de streaming es similar a la API de streaming para XML (StAX) y está constituida por las interfaces [JsonParser](http://docs.oracle.com/javaee/7/api/javax/json/stream/JsonParser.html) y [JsonGenerator](http://docs.oracle.com/javaee/7/api/javax/json/stream/JsonGenerator.html).

**JsonParser**: Contiene métodos para analizar datos JSON usando el modelo de streaming.

**JsonGenerator**:Contiene métodos para escribir datos JSON en un origen de salida.

En la siguiente tabla se incluyen las clases e interfaces principales de la API de streaming.

Tabla 2. Clases principales de la API de streaming

|  |  |
| --- | --- |
| Clase o interfaz | Descripción |
| Json | Contiene métodos estáticos para crear analizadores y generadores JSON, y sus objetos de fábrica. |
| JsonParser | Representa un analizador basado en eventos que puede leer datos JSON en un stream. |
| JsonGenerator | Escribe datos JSON en forma de stream, con un valor por vez. |

JsonParser proporciona acceso directo de solo lectura a datos JSON usando el modelo de programación con análisis pull. En este modelo, el código de aplicación controla el subproceso y llama métodos en la interfaz del analizador para hacer avanzar el analizador u obtener datos JSON del estado actual del analizador.

JsonGenerator proporciona métodos para escribir datos JSON en un stream. El generador puede usarse para escribir pares de nombres/valores en objetos JSON y valores en matrices JSON.   
La API de streaming es una API de bajo nivel diseñada para procesar grandes cantidades de datos JSON de manera eficiente. Con esta API, es posible implementar otros marcos JSON (por ejemplo, enlaces JSON).   
A continuación, utilizaremos la API de streaming con el mismo fin que usamos la API de modelos de objetos, es decir, llevar a cabo una búsqueda de posts públicos de Facebook sobre java. En el Listado 4, las líneas 1 a 3 crean un analizador de streaming; las líneas 4 a 5 obtienen el evento siguiente; la línea 6 busca el evento KEY\_NAME; las líneas 8 a 11 leen los nombres y los imprimen; y las líneas 14 a 16 leen los posts públicos y los imprimen. El uso de la API de streaming proporciona un modo eficiente de obtener acceso a nombres y sus posts públicos si se compara con el uso de la API de modelos de objetos para llevar a cabo la misma tarea.

URL url = new URL("https://graph.facebook.com/search?q=java&type=post");

try (InputStream is = url.openStream();

      JsonParser parser = Json.createParser(is)) {

     while (parser.hasNext()) {

         Event e = parser.next();

         if (e == Event.KEY\_NAME) {

             switch (parser.getString()) {

                 case "name":

                     parser.next();

                    System.out.print(parser.getString());

                    System.out.print(": ");

                    break;

                case "message":

                    parser.next();

                    System.out.println(parser.getString());

                   System.out.println("---------");

                    break;

             }

         }

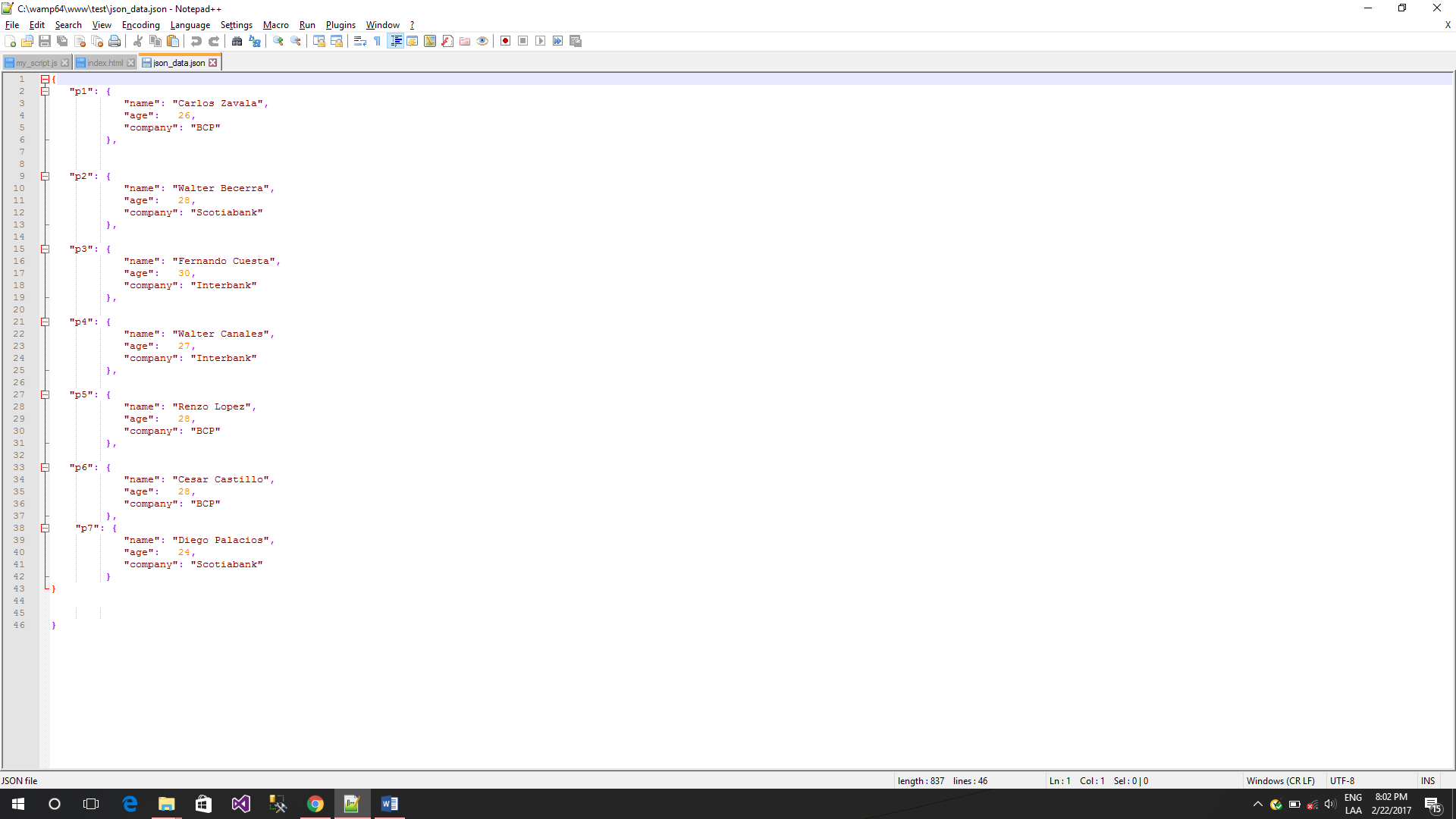
     }

}

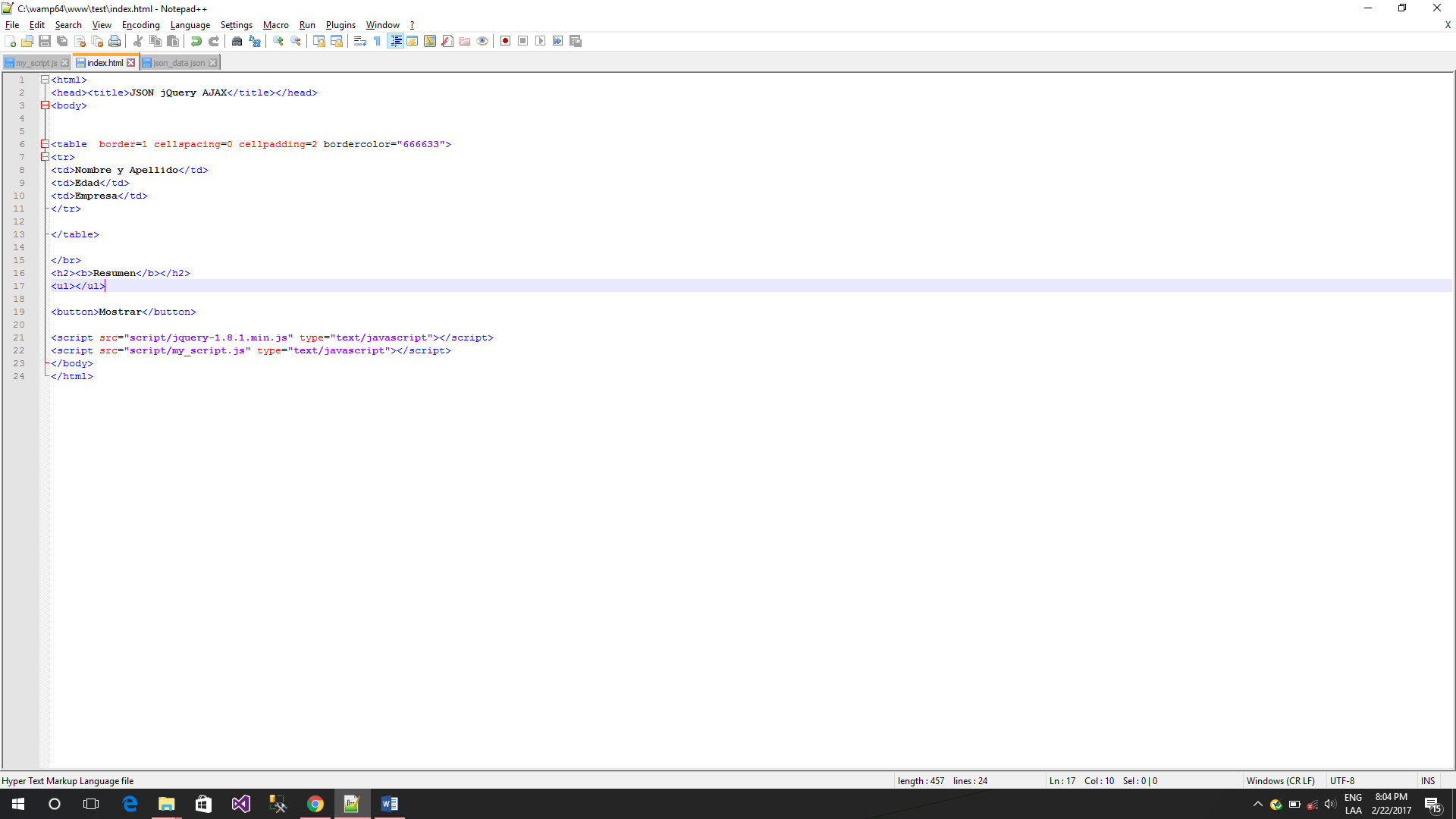
Listado 4. Procesamiento de posts de Facebook con la API de streaming

**6.Caso demostrativo**

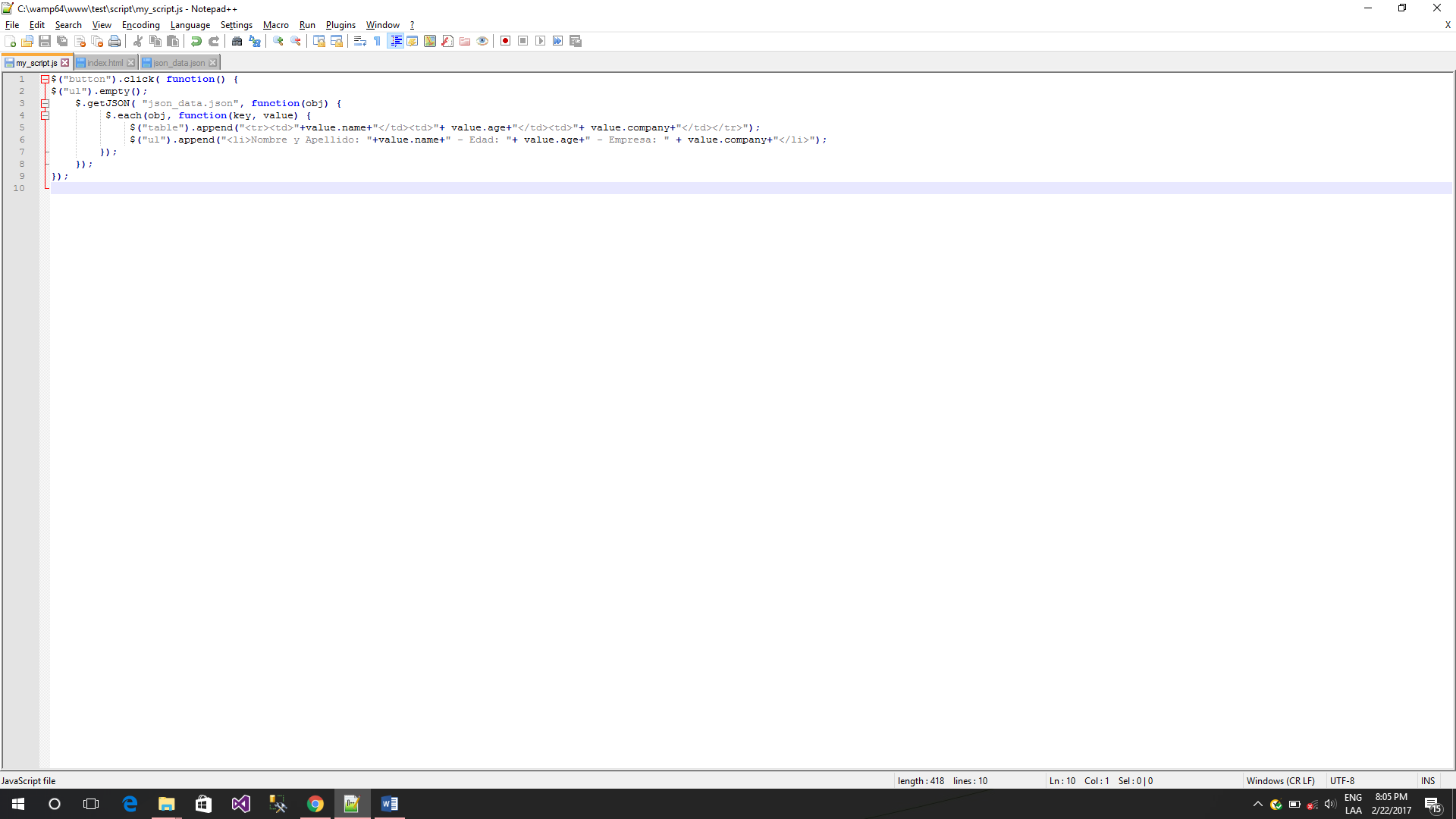
a. Tenemos el archivo “json\_data.json” con la siguiente informacion:



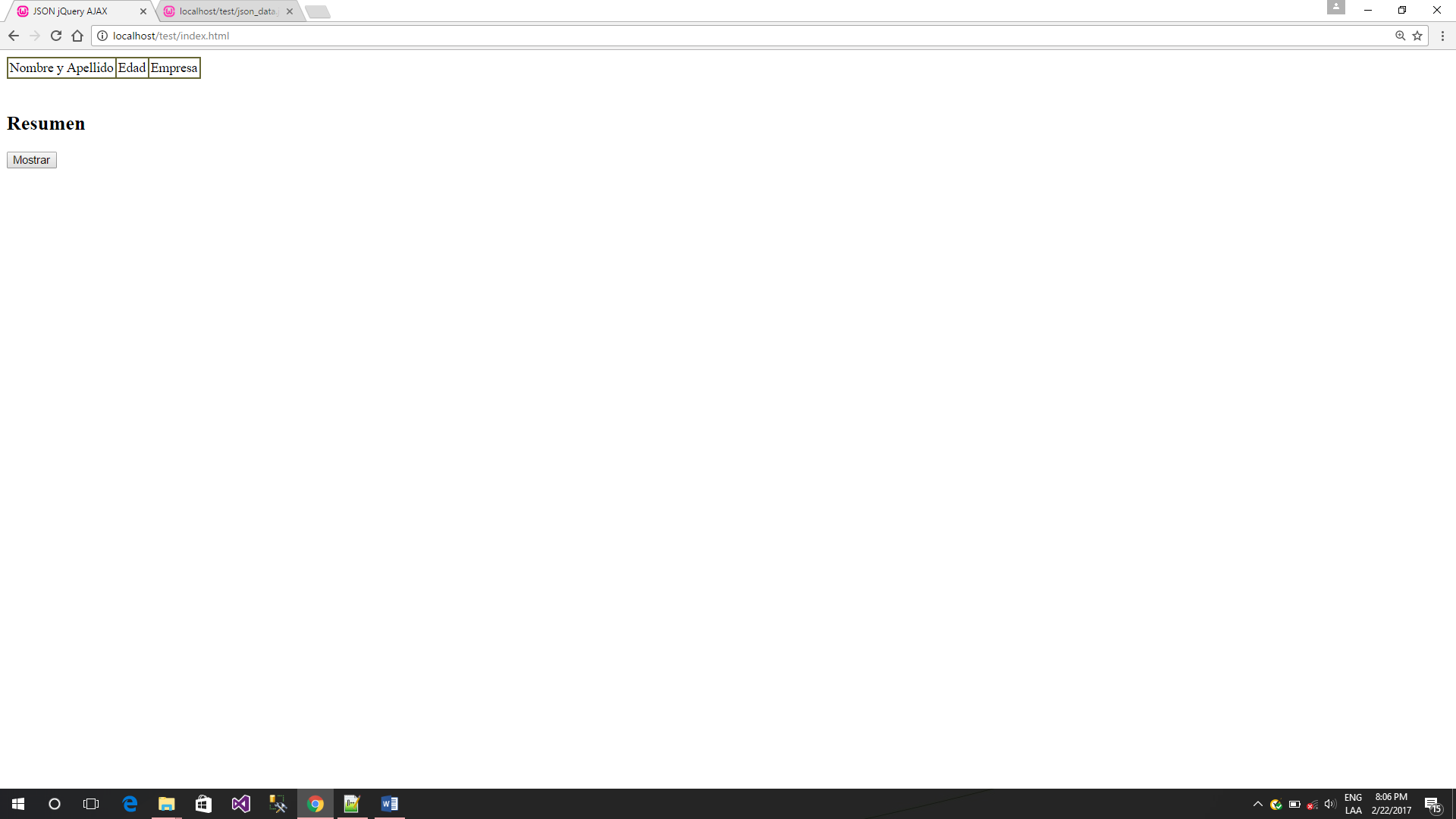
b) Luego tenemos el archivo “index.html” el cual jalara los archivos “js” (1 sera el JavaScript preparado por nosotros y el otro será la librería de JQuery)



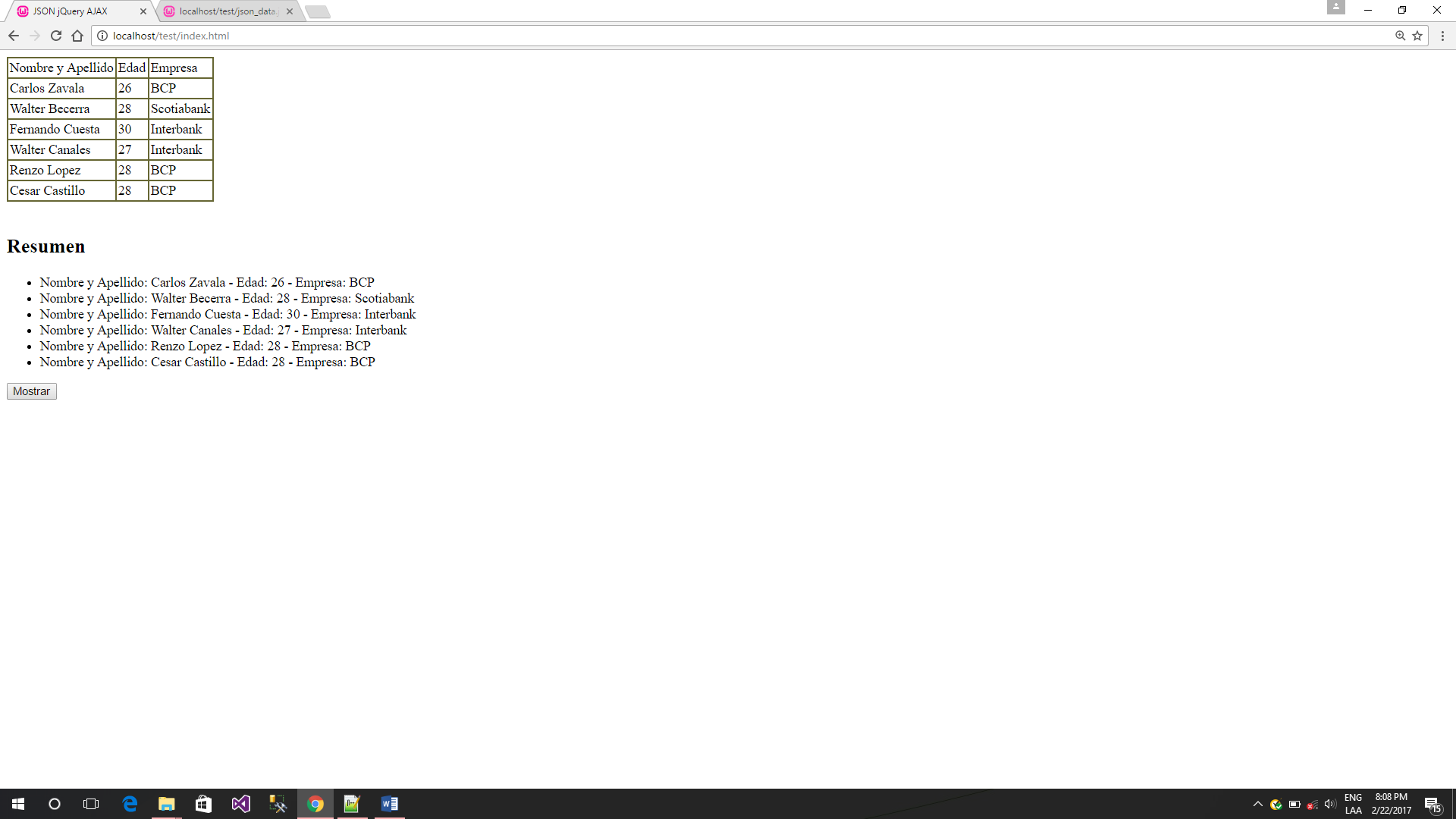
c) Por ultimo tenemos nuestro archivo “my\_script.js” el cual almacenara nuestra función en JavaScript, el cual se ejecutara al hacer clic en el botón del archivo “index”



d) finalmente la página que nos cargara será la siguiente:



Y al hacer clic en el botón “Mostrar” nos mostrara los datos almacenados en el archivo “json” ordenados en una etiqueta “<table>” y en una etiqueta “<ul>”



**7.Discusion.**

En este punto, teníamos un concepto diferente no estábamos de acuerdo con el tema de antecedentes pero con ayuda del profesor que nos mostró en forma ilustrativa como llegó JSON, nos quedó claro

JSON nace por que XML es muy pesado al realizar el intercambio de información

JSON utilizaba librerías de simplejson.jar, Gson.jar (google), y ahora utiliza su propia librería API JSON de Java.

Por lo tanto,

API JSON de Java en muchas aplicaciones reemplaza al XML por ser mas liviano al momento de realizar la transferencia de datos vía remota con la facilidad de trabajar y entender los datos por utilizar un formato de JavaScrip

**8.Conclusiones**

La API de Java para procesamiento JSON proporciona las capacidades siguientes:

* Analiza streams de entrada en objetos inmutables o streams de eventos
* Escribe streams de eventos u objetos inmutables como streams de salida
* Navega mediante programación objetos inmutables
* Construye mediante programación objetos inmutables con generadores

La API sirve como base para desarrollar rutinas API de transformación, consulta, enlace u otras manipulaciones de datos. JAX-RS 2.0 suministra integración nativa para la API de Java para procesamiento JSON.

Para manejos de grandes datos y con mayor seguridad se utiliza SOAP para XML, se hace muy pesado la transferencia de datos

Para maneo de pequeños datos y con mayor velocidad y fácil de entender se utiliza REST con JSON.

**9 Recomendaciones**

SOAP (Simple Object Access Protocol) Protocolo estándar de acceso

XML es mucho mejor para la comunicación entre servidores y aplicaciones ya que permite mensajes mucho mas completos, tipificados y ademas pueden ser validados con facilidad.

REST (Representational State Transfer)Transferencia de Estado Representacional.

JSON cuenta con un formato mas simple y fácil de procesar por lo que es ideal para dispositivos de bajo nivel de procesamiento como los teléfonos móviles los cuales son beneficiados ya que se requieren de menos consumo de datos para descargarlos y un consumo menor de batería al procesarlos.

Desde luego que los teléfonos modernos ya son capaces de procesar información mucho mas rápido pero no podemos confiarnos en todo mundo tendrá uno de esos.

**10 Bibliografía**

* <http://es.wikipedia.org/wiki/JSON>
* <http://json.org/json-es.html>
* <http://json.org/>
* Protocolo REST-Json.pdf
* Jeetlibro.pdf