**Juan Camilo Restrepo Velez 000373886**

**1. API REST para conexión a MariaDB**

Es una interfaz de programación de aplicaciones que se apoya en la arquitectura REST para el desarrollo de aplicaciones en red para obtener datos o generar operaciones sobre esos datos en formatos como XML y JSON, es una alternativa a otros protocolos estándar de intercambio de datos como SOAP. Son cuatro las operaciones que más se usan en la API REST:

* GET (consulta y lectura)
* POST (crear)
* DELETE (eliminar)
* PUT (editar)

**2. Consumir un API**

**AUTOMATIC API REST**

Es una herramienta Open Source bajo licencia Apache 2.0 que te permite crear una completa API de tu base de datos MySQL o MariaDB. Automatic API Rest proporciona una red automática de enlaces proporcionándote la información en formato JSON).

**Instalación**

1. Descargar Automatic API Rest desde [aquí](http://goo.gl/7kPWWP).
2. Extraer el zip y subir todo el directorio a la raíz principal al servidor (en nuestro caso al nodo Raspberry).
3. Abrir el archivo config.php y rellenar todos los campos con las credenciales de la base de datos.
4. Visitar la dirección http://[DireccionIP o Dominio]/AutomaticaApiRest

**Consumir**

Si todo sale bien, saldrá una ventana de LogIn, que es el usuario y contraseña establecido en el archivo config.php (por defecto el usuario y contraseña es admin).

Al entrar nos mostrará una ventana para gestionar las tablas donde nos muestra el nombre de las tablas (Table Name), la dirección donde se puede consultar todos los datos en formato JSON de la tabla (API Link), mostrar la tabla con los datos (Show Table) y la privacidad de la tabla (Privacity).

Al presionar sobre una tabla, nos llevará a otra ventana que nos permitirá gestionar la tabla seleccionada que nos muestra el nombre de las columnas (Column), un enlace que genera un JSON con toda la información del campo (API Link), también mostrar la tabla con los datos del campo (Show Table), la opción de una consulta personalizada (Select) y la privacidad de los campos (Privacity).

Entonces para consultar un campo desde código, lo que hay que hacer es obtener un JSON por medio del API LINK que deseamos y descodificar el JSON para mostrar los campos que nos interesa.

Más información en: <https://geekytheory.com/automatic-api-rest>

**API para procesar JSON**

JSON (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.

Un objeto es una colección no ordenada de cero o más pares de nombres/valores. Una matriz es una secuencia ordenada de cero o más valores. Los valores pueden ser cadenas, números, booleanos, nulos y estos dos tipos estructurados.

Ejemplo de un objeto persona 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"

        }

    ]

}

Dado que en API REST devuelve JSON tenemos que poder procesarlos desde código Java, que para este hay dos API’s: API de modelos de objetos y API de streaming.

**API de modelos de objetos**

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 y JsonArray.

Las siguientes son las clases principales de la API

|  |  |
| --- | --- |
| **Clase** | **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. |
| JsonGenerator | Escribe datos JSON en forma de stream, con un valor por vez. |

El siguiente es un ejemplo de procesar un JSON de los post realizados en Facebook

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(); // Para los resultados

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

     for (JsonObject result : results.getValuesAs(JsonObject.class)) { //Itera sobre cada resultado

         System.out.print(result.getJsonObject("from").getString("name")); //Obtener el nombre de la persona

         System.out.print(": ");

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

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

     }

}

El JSON tiene el siguiente formato

{

     "data" : [

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

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

         ...

     ],

     ...

}

**API de streaming**

Es una API de bajo nivel diseñada para procesar grandes cantidades de datos JSON de manera eficiente y está constituida por las interfaces JsonParser y JsonGenerator. JsonParser contiene métodos para analizar datos JSON usando el modelo de streaming. JsonGenerator contiene métodos para escribir datos JSON en un origen de salida.

Los siguientes son las clases principales de 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. |

El siguiente es un ejemplo de implementación en Java para el mismo fin que el ejemplo con el API de modelos de objetos

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": //Leer nombres

                     parser.next();

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

                    System.out.print(": ");

                    break;

                case "message": //Leer post

                    parser.next();

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

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

                    break;

             }

         }

     }

}

**3. Implementar un API**

Los siguientes enlaces son ejemplos y guías para crear un API REST desde cero.

<http://arieloliva.com/api-rest-bd-mysql/>

<https://www.youtube.com/watch?v=s_ht4AKnWZg>

<https://www.youtube.com/watch?v=Xwbv8sCX4mI>

**Referencias**

¿Qué es Api de rest y para que sirve? - Neo Wiki | NeoAttack. Neoattack. Retrieved 9 April 2020, from <https://neoattack.com/neowiki/api-de-rest/>

API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos. BBVAOpen4U. Retrieved 9 April 2020, from <https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos>

Theory, G. Automatic API Rest: Interactúa con tu base de datos en cualquier plataforma. Geeky Theory. Retrieved 9 April 2020, from <https://geekytheory.com/automatic-api-rest>

API de Java para procesamiento JSON: Introducción a JSON. Oracle.com. (2020). Retrieved 9 April 2020, from <https://www.oracle.com/technetwork/es/articles/java/api-java-para-json-2251318-esa.html>