Arturo Wong
Lic. Ciencias de la Informática
20-12-2020

Contenido

Documentación de AlboTest

En este documento presenta el código fuente utilizado para la generación de la prueba de Albo con la API de Marvel. Contiene los códigos fuentes de App.java, los index.php para los servicios.

El código de app.java contiene diversos métodos:

Método	Parámetros	Función
getGreeting		Función generada por el gradle de manera original
readAll	Reader rd	Función de llamada para construir el JSON mediante la lectura del texto de la URL y el Buffered Reader
readJsonFromUrl	String url	Función de llamado a para leer un JSON desde una URL lo convierte en JSONObject para manejo
callIronManCreators		Función de llamado a ENDPOINT que regresa el JSON con los creadores por el ID de Iron Man ID 1009368
callCapAmericaCreators		Función de llamado a ENDPOINT que regresa el JSON con los creadores por el ID de capitán América ID 1009220
callCharactersFromComic	String comicNo	Función de llamado a ENDPOINT que regresa el JSON con los personajes por ID de comic
aMySQLConnection	String url String user String pass	Función principal para la conexión a la base de datos requiere los parámetros de conexión MySQL

getHerosData		Función que consulta la base de datos para obtener los detalles de id de Marvel y del id de comic
getHerosDataCreators		Función que consulta la base de datos para obtener la tabla de creadores ambos héroes
getHerosDataOtherHeros		Función que consulta la base de datos para obtener la tabla de creadores de ambos héroes
heroClearCreators	String hero_rel	Función que borra a todos los creadores de la tabla de creadores por el héroe relacionado que se recibe como parámetro puede ser 1 IronMan 2 CapAmerica
heroUpdateCreators	String name String rol String id_hero_rel	Función que recibe parámetros de nombre, rol, y héroe relacionado para insertar en la tabla de creadores.
otherHerosClear	String hero_rel	Función que borra a todos los héroes relacionados de la tabla de other_heros por id del héroe relacionado que se recibe como parámetro ser 1- IronMan 2- 2-CapAmerica
otherHerosUpdate	String other_hero String comic_name String id_hero_rel	Función que recibe parámetros de nombre del héroe que sale junto con el principal, el nombre de comic donde aparece, y héroe relacionado para insertar en la tabla de otros héroes.
lastSyncUpdate	String id_hero	Función que actualiza la última sincronización de los datos por el id de héroe
main	String[] args	Función principal que se ejecuta al correr el gradlew run, manda a llamar a todos los métodos internos de la clase para ejecución de los mismos

Arturo Wong
Lic. Ciencias de la Informática
20-12-2020

Código de ejecución del gradle de App.java

```
/*
 * Esta clase manda a llamar a los métodos para traer la informacion de Iron Man y Capitán América, trae a los creadores y el comic modificado más reciente de ese héroe por medio de ENDPOINTS dados por la API de Marvel, así mismo manda a llamar los datos de los héroes que salen en ese comic relacionados.
 */
package albopackage;


//Importe de librerías de java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URL;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.UnknownHostException;
import java.util.concurrent.TimeUnit;
//Importe de librerías de org
//Importe de librerías de entrada de datos de java
import java.io.InputStream;
import java.io.Reader;
import java.nio.charset.Charset;
//Importe de librerías declaradas en la dependencia del gradle
import org.json.*;
import org.json.JSONException;
import org.json.JSONObject;
//Importe de librerías de conexión BD y manejo de resultsets
import java.sql.ResultSet;
```

Arturo Wong
Lic. Ciencias de la Informática
20-12-2020

```
import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.sql.Statement;

import java.sql.Connection;

import java.sql.DriverManager;


public class App {


   /**
    * Función generada por el gradle de manera original
    */
   public String getGreeting() {
      return "Hello World!";
   }


   /**
    * Función de llamada para construir el JSON mediante la lectura del texto
    * de la URL y el Buffered Reader
    *
    * @param Reader rd
    */
   private static String readAll(Reader rd) throws IOException {
      StringBuilder sb = new StringBuilder();
      int cp;
      while ((cp = rd.read()) != -1) {
         sb.append((char) cp);
      }
      return sb.toString();
   }
```

```
  /**
   * Función de llamado a para leer un JSON desde una URL lo convierte en
   * JSONObject para manejo
   *
   * @param String url
   */
  public static JSONObject readJsonFromUrl(String url) throws IOException, JSONException {
    InputStream is = new URL(url).openStream();
    try {
      BufferedReader rd = new BufferedReader(new InputStreamReader(is,
Charset.forName("UTF-8")));
      String jsonText = readAll(rd);
      JSONObject json = new JSONObject(jsonText);
      return json;
    } finally {
      is.close();
    }
  }

  /**
   * Función de llamado a ENDPOINT que regresa el JSON con los creadores por
   * el ID de Iron Man ID 1009368
   */
  public static void callIronManCreators() throws MalformedURLException, IOException,
SQLException {
    //Se obtiene la respuesta JSON de la url para los datos
    JSONObject jsonIM = null;
    try {
```

```
        jsonIM =
readJsonFromUrl("https://gateway.marvel.com/v1/public/characters/1009368/comics?orderBy=-
modified&apikey=ac50fd1d19c4f4e2727b3444951a8573&hash=fab68ac6420bf936b28e040b1d24
d9bd&ts=1&limit=1");
    } catch (IOException ioex) {
       System.out.println("Error en la lectura de la API de MARVEL");
       return;
    }
    //ENDPOINT que da el id del HEROE =
https://gateway.marvel.com/v1/public/characters?nameStartsWith=iron%20man&apikey=ac50fd
1d19c4f4e2727b3444951a8573&hash=fab68ac6420bf936b28e040b1d24d9bd&ts=1
    //ID iron man = 1009368
    try {
       System.out.println("----------");
       System.out.println("ID iron man = 1009368");
       JSONArray arr = jsonIM.getJSONObject("data").getJSONArray("results");
       System.out.println("Id Comic:" + arr.getJSONObject(0).getInt("id"));
       System.out.println("----------");
       JSONArray creatorsArr =
arr.getJSONObject(0).getJSONObject("creators").getJSONArray("items");
       System.out.println("Creadores");
       try {
          heroClearCreators("1");
       } catch (SQLException w) {
          System.out.println("Error:");
          System.out.println(w);
       }
       for (int i = 0; i < creatorsArr.length(); i++) {
          try {
             heroUpdateCreators(creatorsArr.getJSONObject(i).getString("name"),
creatorsArr.getJSONObject(i).getString("role"), "1");
```

```
            } catch (SQLException w) {
               System.out.println("Error:");
               System.out.println(w);
            }
            System.out.println("Nombre: " + creatorsArr.getJSONObject(i).getString("name"));
            System.out.println("Rol   : " + creatorsArr.getJSONObject(i).getString("role"));
         }
         otherHerosClear("1");
         callCharactersFromComic(String.valueOf(arr.getJSONObject(0).getInt("id")), "1");
         lastSyncUpdate("1");
         System.out.println("----------");
      } catch (NullPointerException npex) {
         System.out.println("Error por apuntador nulo en callIronManCreators");
         return;
      }
      //return false;
   }

   /**
    * Función de llamado a ENDPOINT que regresa el JSON con los creadores por
    * el ID de capitán América ID 1009220
    */
   public static void callCapAmericaCreators() throws MalformedURLException, IOException,
SQLException {
      //Se obtiene la respuesta JSON de la url para los datos
      JSONObject jsonCap = null;
      try {
         jsonCap =
readJsonFromUrl("https://gateway.marvel.com/v1/public/characters/1009220/comics?orderBy=-
```

```
modified&apikey=ac50fd1d19c4f4e2727b3444951a8573&hash=fab68ac6420bf936b28e040b1d24
d9bd&ts=1&limit=1");

    } catch (IOException ioex) {

       System.out.println("Error en la lectura de la API de MARVEL");

       return;

    }

    //ENDPOINT que da el id del HEROE =
https://gateway.marvel.com/v1/public/characters?nameStartsWith=Captain%20America&apikey=
ac50fd1d19c4f4e2727b3444951a8573&hash=fab68ac6420bf936b28e040b1d24d9bd&ts=1

    //ID capitán América = 1009220

    //ID del comic

    try {

       //Se obtiene el total de resultados dentro del tag del JSON total

       int loopComicsCAP = jsonCap.getJSONObject("data").getInt("total");

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

       System.out.println("ID capitán América = 1009220");

       //Se recorre el arreglo de resultados para

       JSONArray arr = jsonCap.getJSONObject("data").getJSONArray("results");

       System.out.println("Id Comic:" + arr.getJSONObject(0).getInt("id"));

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

       JSONArray creatorsArr =
arr.getJSONObject(0).getJSONObject("creators").getJSONArray("items");

       System.out.println("Creadores");

       try {

          heroClearCreators("2");

       } catch (SQLException w) {

          System.out.println("Error:");

          System.out.println(w);

       }

       for (int i = 0; i < creatorsArr.length(); i++) {
```

```
            try {
                heroUpdateCreators(creatorsArr.getJSONObject(i).getString("name"),
creatorsArr.getJSONObject(i).getString("role"), "2");
            } catch (SQLException w) {
                System.out.println("Error:");
                System.out.println(w);
            }
            System.out.println("Nombre: " + creatorsArr.getJSONObject(i).getString("name"));
            System.out.println("Rol   : " + creatorsArr.getJSONObject(i).getString("role"));
            //System.out.println(arr.getJSONObject(0).getInt("id") + "\n");
          }
          otherHerosClear("2");
          callCharactersFromComic(String.valueOf(arr.getJSONObject(0).getInt("id")), "2");
          lastSyncUpdate("2");
          System.out.println("----------");
      } catch (NullPointerException npex) {
          System.out.println("Error por apuntador nulo en callCapAmericaCreators");
          return;
      }
  }

  /**
   * Función de llamado a ENDPOINT que regresa el JSON con los personajes por
   * el ID de comic
   *
   * @param String comicNo
   */
  public static void callCharactersFromComic(String comicNo, String id_hero_related) throws
MalformedURLException, IOException, SQLException {
```

```
        JSONObject jsonIM = null;

        try{

           jsonIM = readJsonFromUrl("https://gateway.marvel.com/v1/public/comics/" + comicNo +
"/characters?orderBy=name&apikey=ac50fd1d19c4f4e2727b3444951a8573&hash=fab68ac6420b
f936b28e040b1d24d9bd&ts=1");

        } catch (IOException ioex) {

           System.out.println("Error en la lectura de la API de MARVEL");

           return;

        }

        //JSONObject jsonIM = readJsonFromUrl("https://gateway.marvel.com/v1/public/comics/" +
comicNo +
"/characters?orderBy=name&apikey=ac50fd1d19c4f4e2727b3444951a8573&hash=fab68ac6420b
f936b28e040b1d24d9bd&ts=1");

        try{

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

        JSONArray arr = jsonIM.getJSONObject("data").getJSONArray("results");

        System.out.println("Characters in comic " + comicNo);

        for (int i = 0; i < arr.length(); i++) {

           System.out.println("Heroe : " + arr.getJSONObject(i).getString("name"));

           JSONArray charactersArr =
arr.getJSONObject(0).getJSONObject("comics").getJSONArray("items");

           System.out.println("Rol   : " + charactersArr.getJSONObject(i).getString("name"));

           try {

              otherHerosUpdate(arr.getJSONObject(i).getString("name"),
charactersArr.getJSONObject(i).getString("name"), id_hero_related);

           } catch (SQLException w) {

              System.out.println("Error:");

              System.out.println(w);

           }

        }

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

```
        }catch( NullPointerException npex){
            System.out.println("Error por apuntador nulo en método de callCharactersFromComic");
            return;
        }
        //return false;
    }

    private static Connection conn;

    /**
     * Función principal para la conexión a la base de datos requiere los
     * parámetros de conexión MySQL
     *
     * @param String url
     * @param String user
     * @param String pass
     */
    public static void aMySQLConnection(String url, String user, String pass) throws SQLException,
ClassNotFoundException {
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn = DriverManager.getConnection(url, user, pass);
        System.out.println("Database connection established.");
    }

    /**
     * Función que consulta la base de datos para obtener los detalles de id de
     * Marvel y el id de comic
     */
    public void getHerosData() throws SQLException {
```

```
        String query = "SELECT * FROM  marvel_repo_db.heros;";
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
        ResultSetMetaData rsmd = rs.getMetaData();

        int columnsNumber = rsmd.getColumnCount();
        while (rs.next()) {
            for (int i = 1; i <= columnsNumber; i++) {
                System.out.print(rs.getString(i) + " ");
            }
            System.out.println();
        }
    }

    /**
     * Función que consulta la base de datos para obtener la tabla de creadores
     * de ambos héroes
     */
    public void getHerosDataCreators() throws SQLException {
        String query = "SELECT * FROM  marvel_repo_db.creators;";
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
        ResultSetMetaData rsmd = rs.getMetaData();

        int columnsNumber = rsmd.getColumnCount();
        while (rs.next()) {
            for (int i = 1; i <= columnsNumber; i++) {
                System.out.print(rs.getString(i) + " ");
            }
```

```
            System.out.println();
        }
    }

    /**
     * Funcion que consulta la base de datos para obtener la tabla de creadores
     * de ambos héroes
     */
    public void getHerosDataOtherHeros() throws SQLException {
        String query = "SELECT * FROM  marvel_repo_db.other_heros;";
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
        ResultSetMetaData rsmd = rs.getMetaData();

        int columnsNumber = rsmd.getColumnCount();
        while (rs.next()) {
            for (int i = 1; i <= columnsNumber; i++) {
                System.out.print(rs.getString(i) + " ");
            }
            System.out.println();
        }
    }

    /**
     * Funcion que borra a todos los creadores de la tabla de creadores por el
     * héroe relacionado que se recibe como parámetro puede ser 1 IronMan 2
     * CapAmerica
     *
     * @param String hero_rel
```

```
    */
   public static void heroClearCreators(String hero_rel) throws SQLException {
      String query = "DELETE FROM marvel_repo_db.creators WHERE id_hero_rel = '" + hero_rel +
"';";
      Statement st = conn.createStatement();
      st.executeUpdate(query);
   }

   /**
    * Funcion que recibe parámetros de nombre, rol, y héroe relacionado para
    * insertar en la tabla de creadores.
    *
    * @param String name
    * @param String rol
    * @param String id_hero_rel
    */
   public static void heroUpdateCreators(String name, String rol, String id_hero_rel) throws
SQLException {
      String query = "INSERT INTO marvel_repo_db.creators (`name`,`rol`,`id_hero_rel`) VALUES ('"
+ name + "', '" + rol + "', '" + id_hero_rel + "');";
      Statement st = conn.createStatement();
      st.executeUpdate(query);
   }

   /**
    * Función que borra a todos los héroes relacionados de la tabla de
    * other_heros por id del héroe relacionado que se recibe como parámetro
    * puede ser 1-IronMan, 2-CapAmerica
    *
    * @param String hero_rel
```

```
    */

   public static void otherHerosClear(String hero_rel) throws SQLException {

      String query = "DELETE FROM marvel_repo_db.other_heros WHERE id_hero_related = '" +
hero_rel + "';";

      Statement st = conn.createStatement();

      st.executeUpdate(query);

   }


   /**

    * Funcion que recibe parámetros de nombre del héroe que sale junto con el

    * principal, el nombre de comic donde aparece, y héroe relacionado para

    * insertar en la tabla de otros héroes.

    *

    * @param String other_hero

    * @param String comic_name

    * @param String id_hero_rel

    */

   public static void otherHerosUpdate(String other_hero, String comic_name, String id_hero_rel)
throws SQLException {

      String query = "INSERT INTO marvel_repo_db.other_heros (`name`,`comic`,`id_hero_related`)
VALUES ('" + other_hero + "','" + comic_name + "','" + id_hero_rel + "');";

      Statement st = conn.createStatement();

      st.executeUpdate(query);

   }


   /**

    * Funcion que actualiza la última sincronización de los datos por el id de

    * héroe

    *

    * @param String id_hero
```

Arturo Wong
Lic. Ciencias de la Informática
20-12-2020

```
    */
   public static void lastSyncUpdate(String id_hero) throws SQLException {
      String query = "UPDATE `marvel_repo_db`.`heros` SET `last_sync` = current_timestamp() WHERE `id_hero` = '" + id_hero + "';";
      Statement st = conn.createStatement();
      st.executeUpdate(query);
   }

   /**
    * Funcion principal que se ejecuta al correr el gradlew run, manda a llamar
    * a todos los métodos internos de la clase para ejecución de los mismos
    */
   public static void main(String[] args) throws MalformedURLException, IOException, InterruptedException, SQLException, ClassNotFoundException {
      //System.out.println(new App().getGreeting());
      try {
         new App().aMySQLConnection("jdbc:mysql://127.0.0.1", "root", "");
      } catch (SQLException sqle) {
         System.out.println("Error en la conexion a la base de datos de MySQL");
         return;
      }
      //Llamado de los metodos para obtener informacion
      new App().getHerosData();
      System.out.println("");
      new App().callIronManCreators();
      new App().callCapAmericaCreators();
      new App().getHerosData();
      System.out.println("");
      new App().getHerosDataCreators();
```

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

        new App().getHerosDataOtherHeros();

        System.out.println("");

        conn.close();

    }

}
```

Índex del servicio de /marvel/characters/capamerica

```
<?php

/* utilizar la variable que nos viene o establecerla nosotros */

@$format = strtolower($_GET['format']) == 'json' ? 'json' : 'xml'; //xml es por defecto

$format = strtolower('json');

@$user_id = intval($_GET['user']);


/* conectamos a la bd */

$link = mysqli_connect('127.0.0.1', 'root', '') or die('No se puede conectar a la BD');


/* sacamos los posts de bd */

$query = "SELECT name, comic FROM  `marvel_repo_db`.`other_heros` WHERE id_hero_related =
'2';";

$result = mysqli_query($link, $query ) or die('Query no funcional:  ' . $query);


/* creamos el array con los datos */

$posts = array();

if (mysqli_num_rows($result)) {

    while ($post = mysqli_fetch_assoc($result)) {

        $posts[] = array('post' => $post);

    }
```

}

```
/* formateamos el resultado */
if ($format == 'json') {
   header('Content-type: application/json');
   echo json_encode(array('posts' => $posts));
} else {
   header('Content-type: text/xml');
   echo '';
   foreach ($posts as $index => $post) {
      if (is_array($post)) {
         foreach ($post as $key => $value) {
            echo '<', $key, '>';
            if (is_array($value)) {
               foreach ($value as $tag => $val) {
                  echo '<', $tag, '>', htmlentities($val), '';
               }
            }
            echo '';
         }
      }
   }
   echo '';
}
/* nos desconectamos de la bd */
@mysqli_close($link);
//}
?>
```

Índex del servicio /marvel/characters/ironman

```
<?php

/* utilizar la variable que nos viene o establecerla nosotros */

@$format = strtolower($_GET['format']) == 'json' ? 'json' : 'xml'; //xml es por defecto

$format = strtolower('json');

@$user_id = intval($_GET['user']);


/* conectamos a la bd */

$link = mysqli_connect('127.0.0.1', 'root', '') or die('No se puede conectar a la BD');


/* sacamos los posts de bd */

$query = "SELECT name, comic FROM  `marvel_repo_db`.`other_heros` WHERE id_hero_related =
'1';";

$result = mysqli_query($link, $query ) or die('Query no funcional:  ' . $query);


/* creamos el array con los datos */

$posts = array();

if (mysqli_num_rows($result)) {

   while ($post = mysqli_fetch_assoc($result)) {

      $posts[] = array('post' => $post);

   }

}


/* formateamos el resultado */

if ($format == 'json') {

   header('Content-type: application/json');

   echo json_encode(array('posts' => $posts));

} else {
```

```
    header('Content-type: text/xml');

    echo '';

    foreach ($posts as $index => $post) {

        if (is_array($post)) {

            foreach ($post as $key => $value) {

                echo '<', $key, '>';

                if (is_array($value)) {

                    foreach ($value as $tag => $val) {

                        echo '<', $tag, '>', htmlentities($val), '';

                    }

                }

                echo '';

            }

        }

    }

    echo '';

}


/* nos desconectamos de la bd */

@mysqli_close($link);


?>
```

Índex del servicio /marvel/colaborators/capamerica

```
<?php

/* utilizar la variable que nos viene o establecerla nosotros */

@$format = strtolower($_GET['format']) == 'json' ? 'json' : 'xml'; //xml es por defecto

$format = strtolower('json');

@$user_id = intval($_GET['user']);
```

```
/* conectamos a la bd */
$link = mysqli_connect('127.0.0.1', 'root', '') or die('No se puede conectar a la BD');

/* sacamos los posts de bd */
$query = "SELECT rol, name FROM  `marvel_repo_db`.`creators` WHERE id_hero_rel = '2';";
$result = mysqli_query($link, $query ) or die('Query no funcional:  ' . $query);

/* creamos el array con los datos */
$posts = array();
if (mysqli_num_rows($result)) {
   while ($post = mysqli_fetch_assoc($result)) {
      $posts[] = array('post' => $post);
   }
}

/* formateamos el resultado */
if ($format == 'json') {
   header('Content-type: application/json');
   echo json_encode(array('posts' => $posts));
} else {
   header('Content-type: text/xml');
   echo '';
   foreach ($posts as $index => $post) {
      if (is_array($post)) {
         foreach ($post as $key => $value) {
            echo '<', $key, '>';
            if (is_array($value)) {
               foreach ($value as $tag => $val) {
```

```
                    echo '<', $tag, '>', htmlentities($val), '';
                }
            }
            echo '';
        }
    }
  }
  echo '';
}

/* nos desconectamos de la bd */
@mysqli_close($link);

?>
```

Índex del servicio /marvel/colaborators/ironman

```
<?php
/* utilizar la variable que nos viene o establecerla nosotros */
@$format = strtolower($_GET['format']) == 'json' ? 'json' : 'xml'; //xml es por defecto
$format = strtolower('json');
@$user_id = intval($_GET['user']);

/* conectamos a la bd */
$link = mysqli_connect('127.0.0.1', 'root', '') or die('No se puede conectar a la BD');

/* sacamos los posts de bd */
$query = "SELECT rol, name FROM  `marvel_repo_db`.`creators` WHERE id_hero_rel = '1';";
$result = mysqli_query($link, $query ) or die('Query no funcional:  ' . $query);
```

```
/* creamos el array con los datos */

$posts = array();

if (mysqli_num_rows($result)) {

   while ($post = mysqli_fetch_assoc($result)) {

      $posts[] = array('post' => $post);

   }

}


/* formateamos el resultado */

if ($format == 'json') {

   header('Content-type: application/json');

   echo json_encode(array('posts' => $posts));

} else {

   header('Content-type: text/xml');

   echo '';

   foreach ($posts as $index => $post) {

      if (is_array($post)) {

         foreach ($post as $key => $value) {

            echo '<', $key, '>';

            if (is_array($value)) {

               foreach ($value as $tag => $val) {

                  echo '<', $tag, '>', htmlentities($val), '';

               }

            }

            echo '';

         }

      }

   }

   echo '';
```

Arturo Wong
Lic. Ciencias de la Informática
20-12-2020
}

/* nos desconectamos de la bd */

@mysqli_close($link);

?>