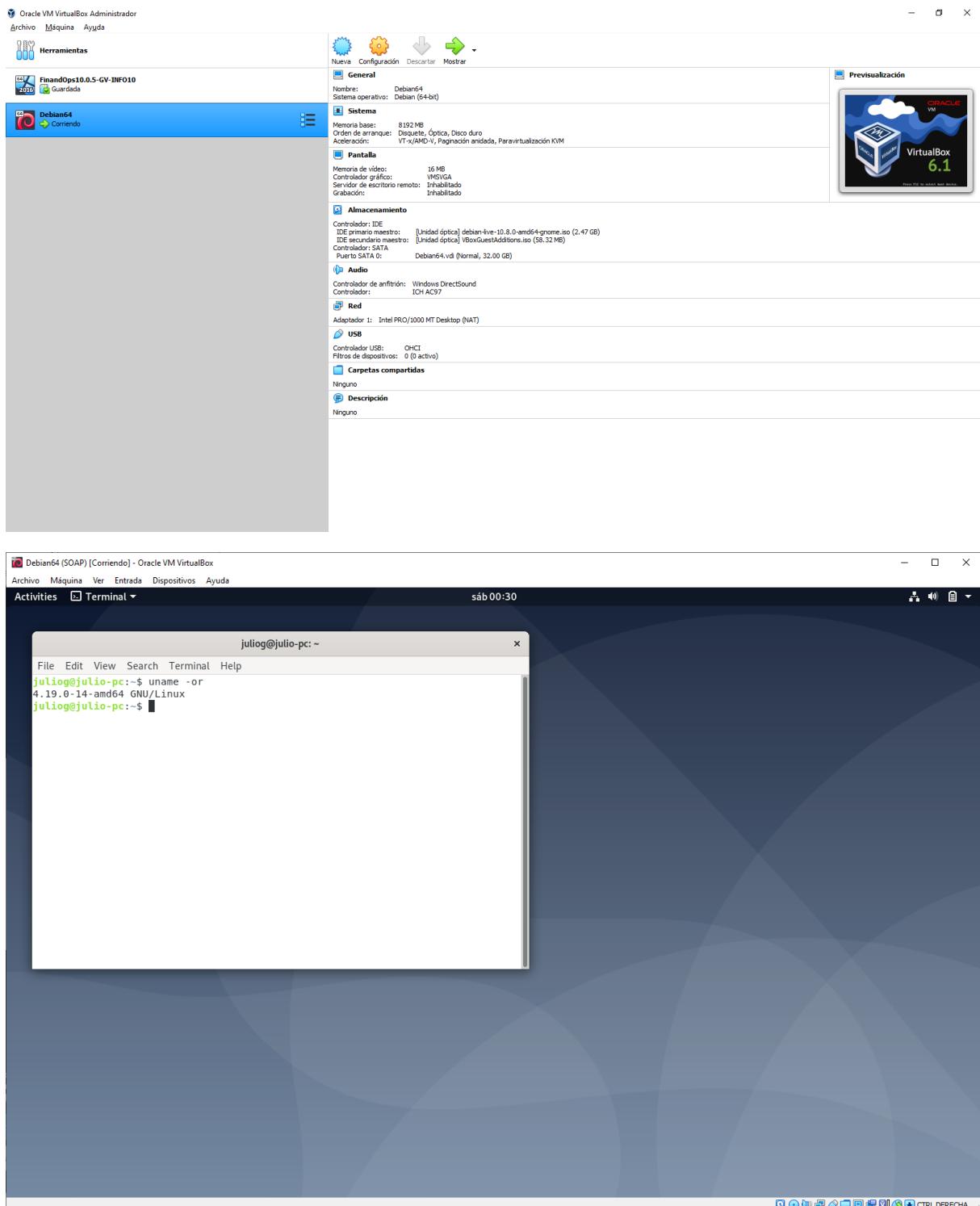
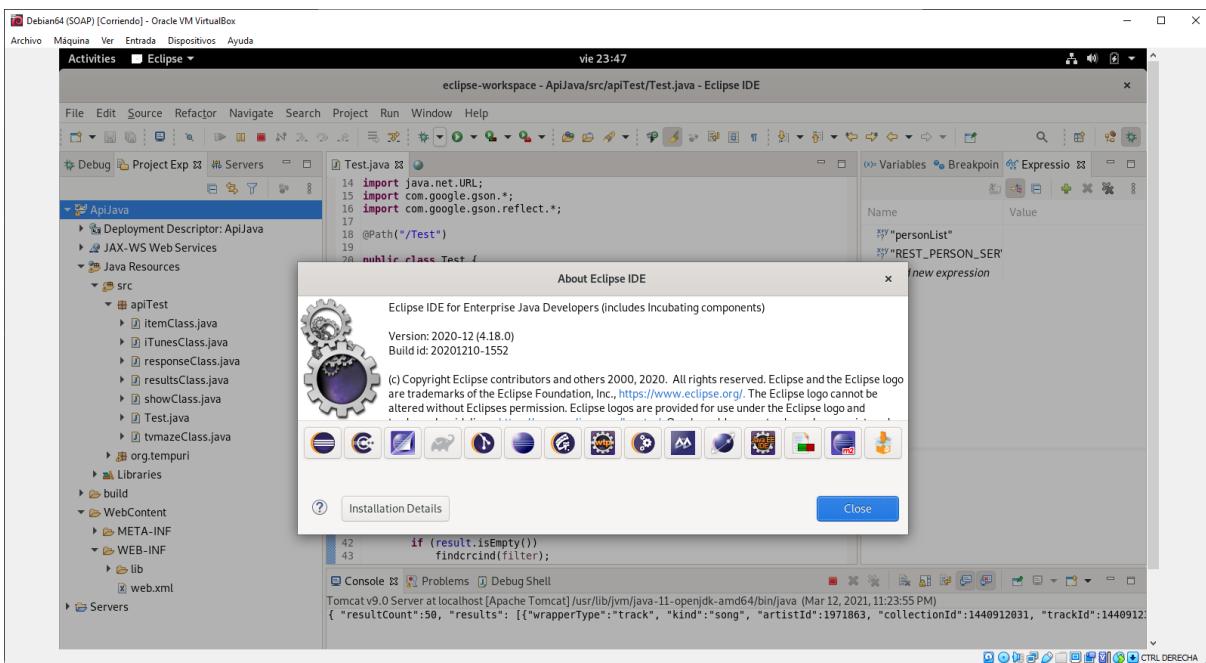


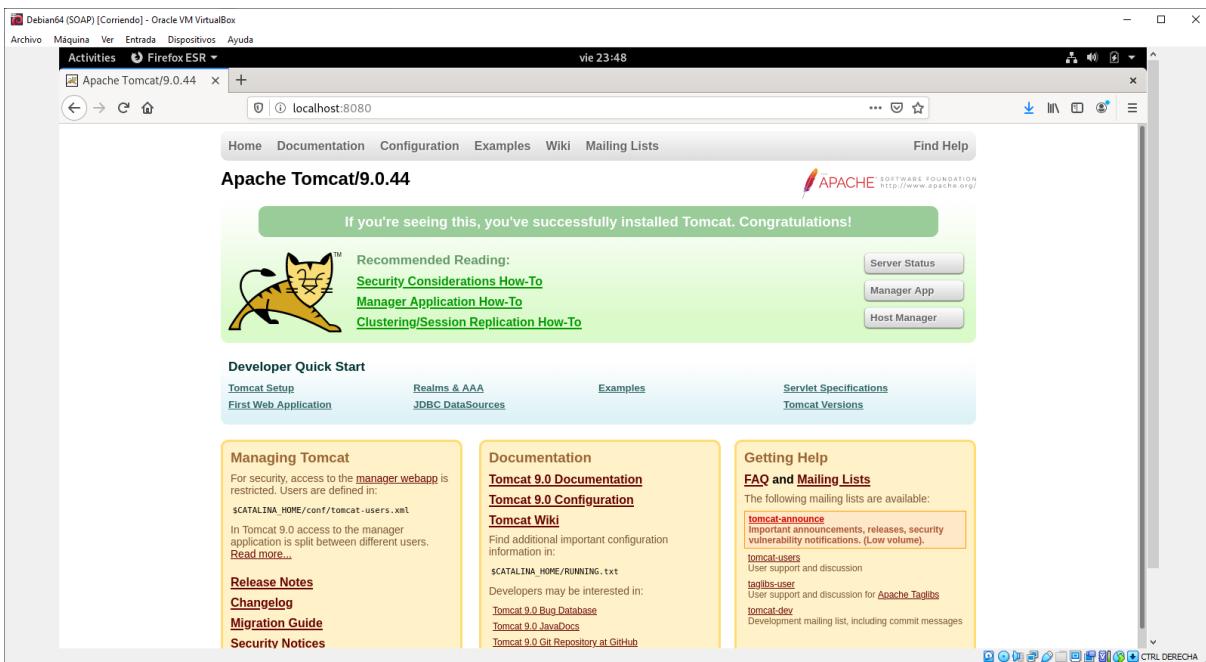
Se utilizó un Linux Debian 10 limpio en una máquina virtual.



Como IDE se utilizó Eclipse



Como servidor de aplicaciones se utilizó Apache Tomcat v9.0.44



Se validó la respuesta de cada una de los 3 orígenes o fuentes de datos.

API-TEST

<http://api.tvmaze.com/search/shows?q=girls> (REST)

Postman

File Edit View Help

Home Workspaces Reports Explore

My Workspace

GET http://api.tvmaze.com/search/shows?q=girls

Params Auth Headers (7) Body Pre-req Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> q	girls	

Cookies Body Cookies Headers (9) Test Results

```

1 {
2   "search": 17.776824,
3   "show": [
4     {
5       "id": 339,
6       "url": "https://www.tvmaze.com/shows/139/girls",
7       "name": "Girls",
8       "language": "English",
9       "genres": [
10         "Drama",
11         "Romance"
12       ],
13       "status": "Ended",
14       "runtime": 30,
15       "premiered": "2012-04-15",
16       "officialSite": "http://www.hbo.com/girls",
17       "schedule": {
18         "time": "22:00",
19         "days": [
20           "Sunday"
21         ]
22       },
23       "rating": {
24         "average": 8.6
25       },
26       "weight": 91,
27       "network": [
28         {
29           "id": 9,
30           "name": "HBO",
31           "country": [
32             {
33               "name": "United States",
34               "code": "US",
35               "timezone": "America/New_York"
36             }
37           ],
38           "webChannel": null,
39           "externals": {
40             ...
41           }
42         }
43       }
44     ]
45   }
46 }

```

Send 200 OK 850 ms 11.61 KB Save Response

Find and Replace Console

<https://itunes.apple.com/search?term=girls> (REST)

Postman

File Edit View Help

Home Workspaces Reports Explore

My Workspace

GET https://itunes.apple.com/search?term=girls

Params Auth Headers (28) Body Pre-req Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> term	girls	

Cookies Body Cookies Headers (28) Test Results

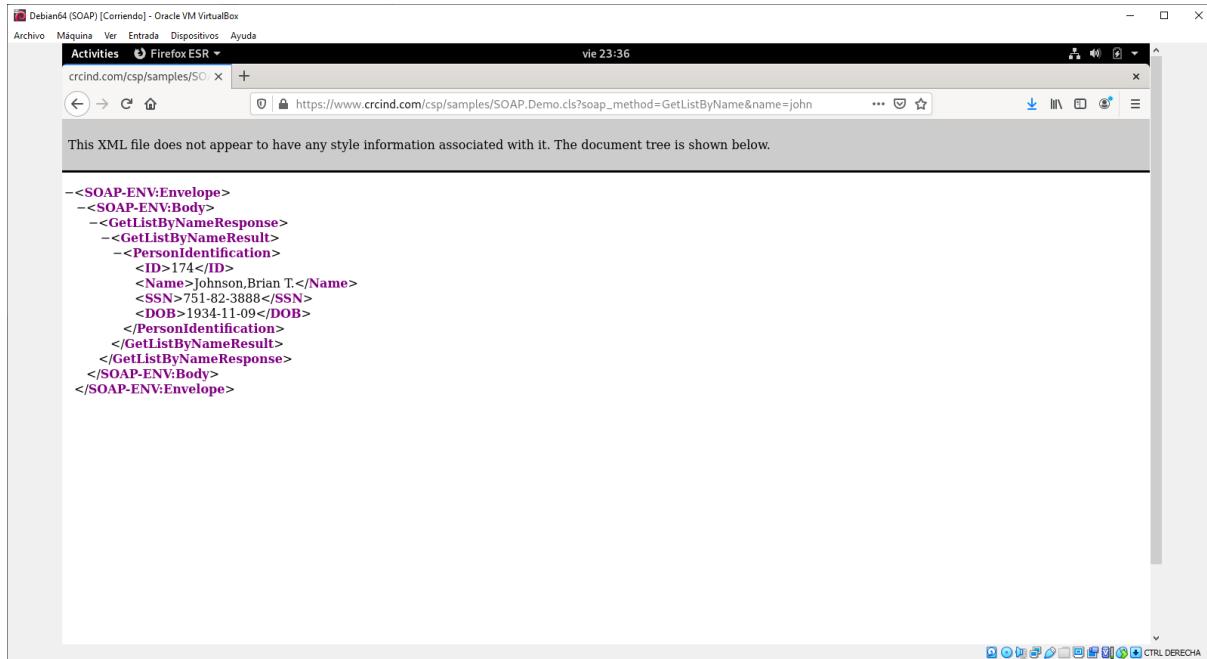
```

1 {
2   "resultCount": 60,
3   "results": [
4     {
5       "resultType": "track",
6       "kind": "song",
7       "artistId": 1971863,
8       "collectionId": 1449912803,
9       "trackName": "Girls",
10      "collectionCensoredName": "Licensed to Ill",
11      "trackCensoredName": "Girls",
12      "artistViewUrl": "https://music.apple.com/us/artist/beastie-boys/1971863",
13      "collectionViewUrl": "https://music.apple.com/us/album/girls-1449912803",
14      "trackViewUrl": "https://music.apple.com/us/track/girls-1449912803",
15      "previewUrl": "https://audio-ssl.itunes.apple.com/itunes-assets/VideoPreview10/v4/0/a/9f/fba9fa2-737a-73d8-914c-6990bdca/ma21013808973167849_plus.acc.p.m4a",
16      "artworkUrl30": "https://is1-ssl-musicstatic.com/image/thumb/mus1114/v4/66/34/0966343e-8f4c-3994-a11b-92cd0bf7540/source/66x34.jpg",
17      "artworkUrl100": "https://is1-ssl-musicstatic.com/image/thumb/mus1114/v4/66/34/0966343e-8f4c-3994-a11b-92cd0bf7540/source/66x66.jpg",
18      "artworkUrl1000": "https://is1-ssl-musicstatic.com/image/thumb/mus1114/v4/66/34/0966343e-8f4c-3994-a11b-92cd0bf7540/source/1000x1000.jpg",
19      "collectionPrice": 9.99,
20      "trackExplicitness": "notExplicit",
21      "collectionExplicitness": "notExplicit",
22      "discCount": 1,
23      "discNumber": 1,
24      "trackCount": 1,
25      "trackNumber": 1,
26      "trackTimeMillis": 132707,
27      "country": "USA",
28      "currency": "USD",
29      "primaryGenreName": "Hip-Hop/Rap",
30      "listenable": true
31    }
32  ]
33 }
34 
```

Send 200 OK 286 ms 14.93 KB Save Response

Find and Replace Console

http://www.crcind.com/csp/samples/SOAP.Demo.cls?soap_method=GetListByName&name=john (SOAP)



La clase principal es Test.java

En esta clase se agregaron las Url para cada servicio para consultar datos

```

25     private String REST_TVMAZE_SERVICE_URL = "http://api.tvmaze.com/search/shows?q=";
26     private String REST_ITUNES_SERVICE_URL = "https://itunes.apple.com/search?term=";
27     private String REST_PERSON_SERVICE_URL = "http://www.crcind.com/csp/samples/SOAP.Demo.cls";
28

```

Se creó un método GET con el nombre `findByName` que recibe como parámetro una cadena para hacer la búsqueda en las 3 fuentes de datos.

```

33     @GET
34     @Path("/find")
35     public Response findByName(@QueryParam("filter") String filter){
36         if (result.isEmpty())
37             findShows(filter);
38
39         if (result.isEmpty())
40             findiTunes(filter);
41
42         if (result.isEmpty())
43             findcrcind(filter);
44
45         if (result.isEmpty())
46             result = PASS;
47
48         response.setResult(result);
49         response.setList(list);
50
51         return response;
52     }

```

Cada búsqueda se realiza utilizando un método diferente (enviando el parámetro `filter`) ya que para buscar shows, canciones y películas el origen de datos es un servicio REST y para buscar personas es un método SOAP.

Para realizar la búsqueda de shows se usa el método `findShows`. Se usa la Url mencionada anteriormente, se indica que es una petición GET y que el tipo de contenido es json. Si el código de respuesta es diferente de 200 (OK) se termina esta

solicitud, de lo contrario se lee completamente la respuesta, la cual es una cadena json y se convierte a una lista de objetos de tipo tvmazeClass.

```

54  private void findShows(String filter) {
55 	try {
56
57  	URL url = new URL(REST_TVMAZE_SERVICE_URL + filter);
58  	HttpURLConnection conn = (HttpURLConnection) url.openConnection();
59  	conn.setRequestMethod("GET");
60  	conn.setRequestProperty("Accept", "application/json");
61  	if (conn.getResponseCode() != 200) {
62  		throw new RuntimeException("Failed : HTTP Error code : "
63  		+ conn.getResponseCode());
64  	}
65  	InputStreamReader in = new InputStreamReader(conn.getInputStream());
66  	BufferedReader br = new BufferedReader(in);
67  	String output;
68  	while ((output = br.readLine()) != null) {
69  		System.out.println(output);
70  		if (output.isEmpty() == false)
71  		{
72  			Gson gson = new Gson();
73
74  			List<tvmazeClass> shows = gson.fromJson(output, new TypeToken<List<tvmazeClass>>() {}.getType());
75
76  			if(shows.isEmpty() == false){
77  				for(tvmazeClass each: shows){
78
79  					//result += each.getShow().getName() + "<br/>";
80
81  					list.add(new itemClass()
82  						.withType("show")
83  						.withName(each.getShow().getName())
84  						.withUrl(each.getShow().getUrl())
85  						.withSummary(each.getShow().getSummary())
86  						);
87
88  				}
89
90 			}
91 			conn.disconnect();
92
93 		} catch (Exception e) {
94 			System.out.println("Exception in NetClientGet:- " + e);
95 			result = FAIL + " - " + e;
96 		}
97 	}
98 }

```

La clase tvmazeClass contiene un objeto de la clase showClass.

```

1  package apiTest;
2
3  public class tvmazeClass {
4  	private float score;
5  	private showClass show;
6
7  	// Getter Methods
8
9  	public float getScore() {
10   	return score;
11  }
12
13  public showClass getShow() {
14   	return show;
15  }
16
17  // Setter Methods
18
19  public void setScore( float score ) {
20   	this.score = score;
21  }
22
23  public void setShow( showClass show ) {
24   	this.show = show;
25  }
26

```

La clase showClass contiene la información que posteriormente se enviara en la respuesta de las coincidencias que cumple el criterio de búsqueda.

```
1 package apiTest;
2
3 public class showClass {
4     private float id;
5     private String url;
6     private String name;
7     private String type;
8     private String summary;
9
10    public float getId() {
11        return id;
12    }
13
14    public String getUrl() {
15        return url;
16    }
17
18    public String getName() {
19        return name;
20    }
21
22    public String getType() {
23        return type;
24    }
25
26    public String getSummary() {
27        return summary;
28    }
29
```

Para realizar la búsqueda de canciones y películas se usa el método `findiTunes`. Se usa la Url mencionada anteriormente, se indica que es una petición `GET` y que el tipo de contenido es `json`. Si el código de respuesta es diferente de `200 (OK)` se termina esta solicitud, de lo contrario se lee completamente la respuesta, la cual es una cadena `json` y se convierte a un objeto de tipo `iTunesClass`.

```

99  private void findiTunes(String filter) {
100 try {
101
102     URL url = new URL(REST_ITUNES_SERVICE_URL + filter);
103     HttpURLConnection conn = (HttpURLConnection) url.openConnection();
104     conn.setRequestMethod("GET");
105     conn.setRequestProperty("Accept", "application/json");
106     if (conn.getResponseCode() != 200) {
107         throw new RuntimeException("Failed : HTTP Error code : "
108             + conn.getResponseCode());
109     }
110     InputStreamReader in = new InputStreamReader(conn.getInputStream());
111     BufferedReader br = new BufferedReader(in);
112
113     StringBuilder output = new StringBuilder();
114     String line;
115
116     while ((line = br.readLine()) != null) {
117         output.append(line);
118     }
119
120     System.out.println(output.toString());
121
122     if (output.toString().isEmpty() == false)
123     {
124         Gson gson = new Gson();
125
126         iTunesClass results = gson.fromJson(output.toString(), new TypeToken<iTunesClass>() {}.getType());
127
128         if(results != null){
129             for(resultsClass each: results.getResults()){
130
131                 //result += each.getResults().getTrackName() + "<br/>";
132
133                 list.add(new itemClass()
134                     .withType(each.getKind())
135                     .withName(each.getTrackName())
136                     .withUrl(each.getPreviewUrl())
137                     .withSummary(each.getLongDescription())
138                 );
139             }
140         }
141     }
142     conn.disconnect();
143
144 } catch (Exception e) {
145     System.out.println("Exception in NetClientGet:- " + e);
146     result = FAIL + " - " + e;
147 }
148 }
```

La clase `iTunesClass` contiene una lista de objetos de la clase `resultsClass`.

```

1 package apiTest;
2
3 import java.util.List;
4
5 public class iTunesClass {
6     private float resultCount;
7     private List<resultsClass> results;
8
9     // Getter Methods
10
11    public float getScore() {
12        return resultCount;
13    }
14
15    public List<resultsClass> getResults() {
16        return results;
17    }
18
19     // Setter Methods
20
21    public void setResultCount( float resultCount ) {
22        this.resultCount = resultCount;
23    }
24
25    public void setResults( List<resultsClass> results ) {
26        this.results = results;
27    }
28
29 }
30

```

La clase `resultsClass` contiene la información que posteriormente se enviará en la respuesta de las coincidencias que cumple el criterio de búsqueda.

```

1 package apiTest;
2
3 public class resultsClass {
4     private float trackId;
5     private String previewUrl;
6     private String trackName;
7     private String kind;
8     private String longDescription;
9
10    public float getTrackId() {
11        return trackId;
12    }
13
14    public String getPreviewUrl() {
15        return previewUrl;
16    }
17
18    public String getTrackName() {
19        return trackName;
20    }
21
22    public String getKind() {
23        return kind;
24    }
25
26    public String getLongDescription() {
27        return longDescription;
28    }
29
30 }
31

```

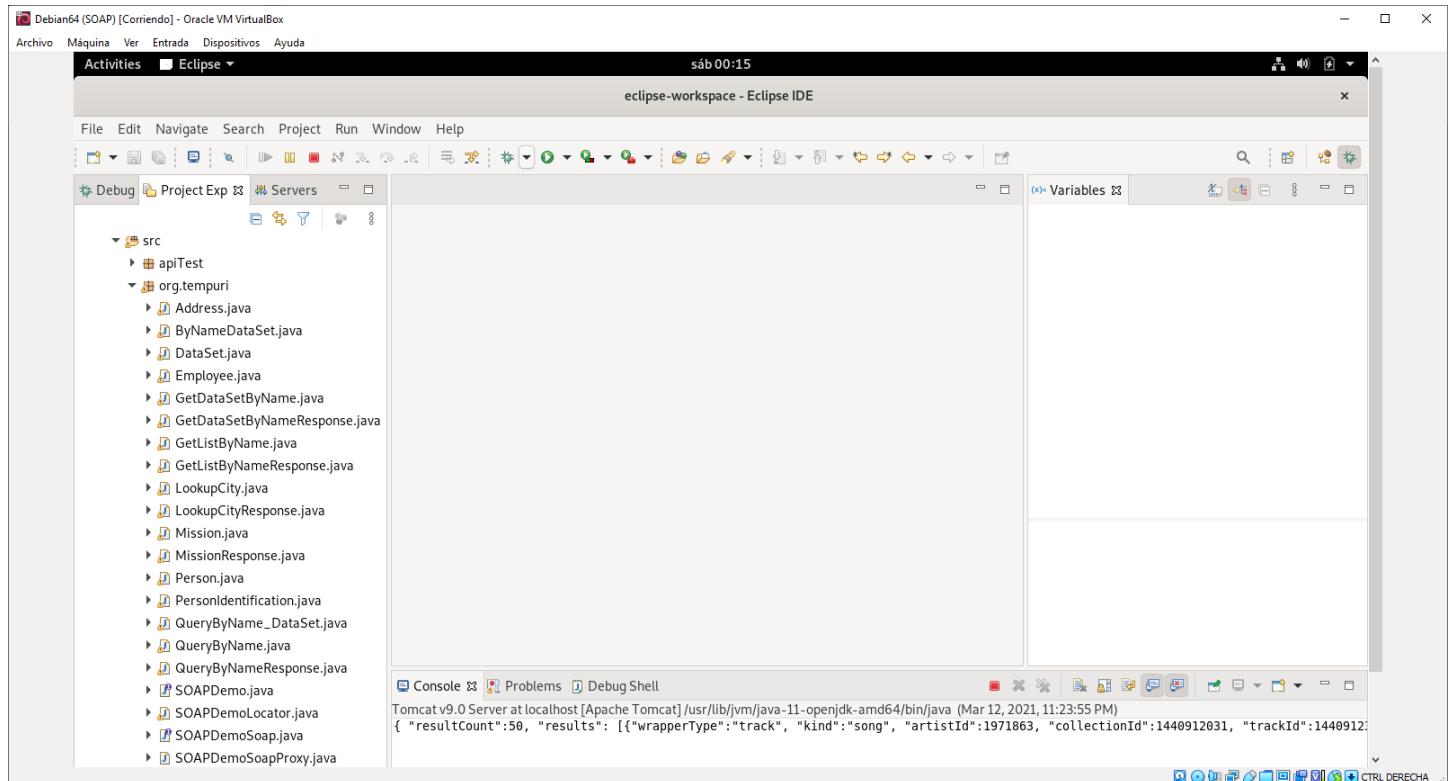
Para el caso del servicio para búsqueda de personas, se hizo la referencia al `WSDL` para que se crearan todas las clases que corresponden.

<http://www.crcind.com/csp/samples/SOAP.Demo.CLS?WSDL=1>

```

<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:s0="http://tempuri.org/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:udl="http://schemas.xmlsoap.org/udl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance" targetNamespace="http://tempuri.org/">
  <types>
    <xs:element name="AddInteger" targetNamespace="http://tempuri.org/">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" name="Arg1" type="s:long"/>
          <xs:element minOccurs="0" name="Arg2" type="s:long"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="AddIntegerResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="AddIntegerResult" type="s:long"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="DivideInteger" targetNamespace="http://tempuri.org/">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" name="Arg1" type="s:long"/>
          <xs:element minOccurs="0" name="Arg2" type="s:long"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="DivideIntegerResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="DivideIntegerResult" type="s:long"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="FindPerson" targetNamespace="http://tempuri.org/">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" name="id" type="s:string"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="FindPersonResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="FindPersonResult" type="s0:Person"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:complexType name="Employee">
      <xs:complexContent>
        <xs:restriction base="s0:Person">
          <xs:sequence>
            <xs:element minOccurs="0" name="Title">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="restriction" type="s:string"/>
                  <xs:element name="maxLength" type="s:int"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:restriction>
      </xs:complexContent>
    </xs:complexType>
  </types>
  <message name="AddIntegerRequest">
    <part name="parameters" type="s0:AddInteger"/>
  
```

Se crearon todos los archivos java que utilizan los métodos y el servicio SOAPDemo.



Para realizar la búsqueda de personas se usa el método `findcrcind`. Se usa la Url mencionada anteriormente, se crea un nuevo cliente `SOAPDemoSoapProxy` y luego se obtiene una instancia `SOAPDemoSoap`. Finalmente se ejecuta el método `getListByName` enviando el parámetro de búsqueda. Si se obtiene respuesta y se tienen coincidencias, se agregan a la lista de resultados.

```

150  private void findcrcind(String filter) {
151      SOAPDemoSoapProxy DemoSoapProxy = new SOAPDemoSoapProxy(REST_PERSON_SERVICE_URL);
152      SOAPDemoSoap DemoSoap = DemoSoapProxy.getSOAPDemoSoap();
153      try {
154          PersonIdentification[] personList = DemoSoap.getListByName(filter);
155
156          if (personList != null)
157          {
158              for(PersonIdentification each: personList)
159              {
160                  //result += each.getName() + "<br/>";
161
162                  list.add(new itemClass()
163                      .withType("person")
164                      .withName(each.getName())
165                      .withUrl(REST_PERSON_SERVICE_URL)
166                      .withSummary("SSN: " + each.getSSN() + " DOB: " + each.getDOB())
167                  );
168              }
169          }
170      } catch (Exception e) {
171          System.out.println(e);
172          result = FAIL + " - " + e;
173      }
174  }
175

```

Para devolver una única lista de resultados se creó la clase `responseClass` que contiene una cadena de respuesta y una lista de coincidencias que corresponden al parámetro de búsqueda.

```

1  package apiTest;
2
3  import java.util.List;
4
5  public class responseClass {
6      private String result;
7      private List<itemClass> list;
8
9      /**
10      * @return the result
11      */
12     public String getResult() {
13         return result;
14     }
15     /**
16     * @param result the result to set
17     */
18     public void setResult(String result) {
19         this.result = result;
20     }
21     /**
22     * @return the list
23     */
24     public List<itemClass> getList() {
25         return list;
26     }
27     /**
28     * @param list the list to set
29     */
30     public void setList(List<itemClass> list) {
31         this.list = list;
32     }
33 }

```

La clase `resultClass` contiene una lista de objetos de la clase `itemClass` que es la información homologada de las 3 fuentes de datos. Los atributos de la clase `itemClass` son:

Type (show, song, tv-episode, person)

Name

Url (es un Url que devuelve la fuente de datos, para el caso de persona siempre se devuelve la Url de la fuente de datos)

Summary (es una descripción del registro que contiene la fuente de datos)

```

1 package apiTest;
2
3 public class itemClass {
4     private String type;
5     private String name;
6     private String url;
7     private String summary;
8
9     /**
10     * @return the type
11     */
12    public String getType() {
13        return type;
14    }
15
16    /**
17     * @param type the type to set
18     */
19    public void setType(String type) {
20        this.type = type;
21    }
22
23    /**
24     * @return the name
25     */
26    public String getName() {
27        return name;
28    }
29
30    /**
31     * @param name the name to set
32     */
33    public void setName(String name) {
34        this.name = name;
35    }
36
37    /**
38     * @return the url
39     */
40    public String getUrl() {
41        return url;
42    }
43
44    /**
45     * @param url the url to set
46     */
47    public void setUrl(String url) {
48        this.url = url;
49    }
50
51    /**
52     * @return the summary
53     */
54    public String getSummary() {
55        return summary;
56    }
57
58    /**
59     * @param summary the summary to set
60     */
61    public void setSummary(String summary) {
62        this.summary = summary;
63    }
64
65    /**
66     * @param type the type to filter
67     */
68    public itemClass withType(String type) {
69        setType(type);
70        return this;
71    }
72
73    /**
74     * @param name the name to filter
75     */
76    public itemClass withName(String name) {
77        setName(name);
78        return this;
79    }
80
81    /**
82     * @param url the url to filter
83     */
84    public itemClass withUrl(String url) {
85        setUrl(url);
86        return this;
87    }
88
89    /**
90     * @param summary the summary to filter
91     */
92    public itemClass withSummary(String summary) {
93        setSummary(summary);
94        return this;
95    }
96
97 }

```

Finalmente se implementó en el servidor de aplicaciones en la dirección <http://localhost:8080/ApiJava/Test/find> y se debe de enviar el parámetro `filter` el criterio de búsqueda.

The screenshot shows the Postman application window. On the left, there's a sidebar with sections for Collections, APIs, Environments, Mock Servers, Monitors, and History. The main area has tabs for Home, Workspaces, Reports, and Explore. A search bar at the top right says "Search Postman". Below it, there are three recent requests: "GET http://localhost:8080...", "GET http://api.tvmaze...", and "GET https://itunes.apple...". The current request is "GET http://localhost:8080/ApiJava/api/Test/find?filter=smith". The "Params" tab is selected, showing a table with a single row: "filter" with value "smith". The "Body" tab shows a JSON response:

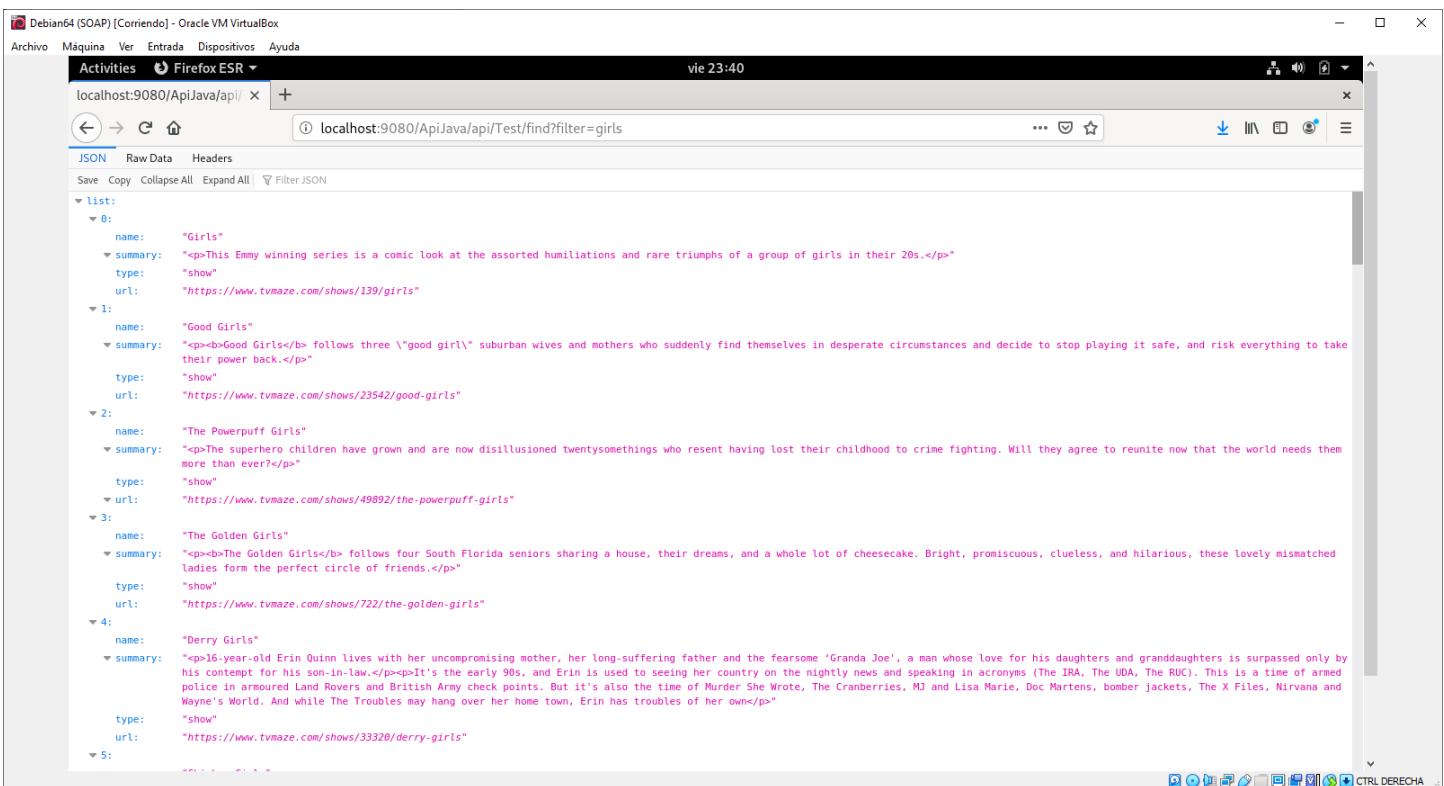
```

1 {
2     "list": [
3         {
4             "name": "Smith",
5             "summary": "<p><b>Smith</b> delves into the world of high stakes robberies, where Bobby Stevens (Liotta) finds himself leading a double life as the leader of a band of thieves who pull off intricate and ingenious jobs and then goes back to being the typical nine-to-five, suburban family man. The series, from executive producer John Wells, creator of 'ER,' will premiere with limited commercial interruption. Leading this close-knit crew of career criminals is Bobby Stevens (Liotta), who appears to be a regular family man with a nine-to-five job, but is actually an expert thief who is seeking just two or three more big paydays so he can finally leave the business for a comfortable, lawful lifestyle with his wife, Hope (Madsen), and their two children. While Bobby never discusses his illegal pursuits, Hope is growing suspicious. Bobby's second family, his core band of partners, each bring their own areas of expertise to pulling off the biggest and most sophisticated armed robberies. Joe (Franky G) is in charge of all things transportation oriented; Jeff (Baker), who is the most

```

Ingresando desde un browser se muestra como la imagen a continuación.

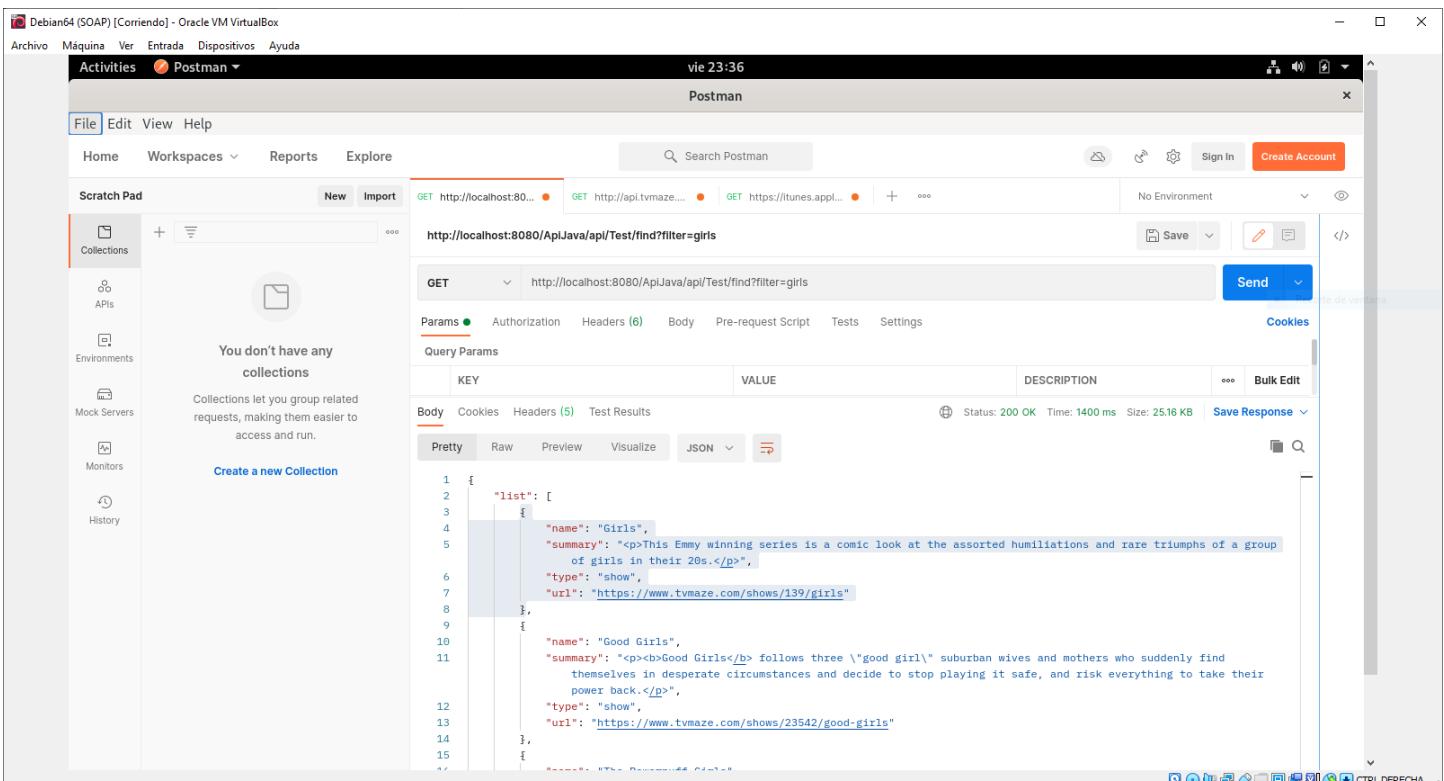
API-TEST



The screenshot shows a Firefox ESR window displaying a JSON response from the URL `localhost:9080/ApiJava/api/Test/find?filter=girls`. The response is a list of five TV show entries, each containing a name, summary, type, and URL. The JSON structure is as follows:

```
list: [
  {
    "name": "Girls",
    "summary": "<p>This Emmy winning series is a comic look at the assorted humiliations and rare triumphs of a group of girls in their 20s.</p>",
    "type": "show",
    "url": "https://www.tvmaze.com/shows/139/girls"
  },
  {
    "name": "Good Girls",
    "summary": "<p><b>Good Girls</b> follows three \"good girl\" suburban wives and mothers who suddenly find themselves in desperate circumstances and decide to stop playing it safe, and risk everything to take their power back.</p>",
    "type": "show",
    "url": "https://www.tvmaze.com/shows/23542/good-girls"
  },
  {
    "name": "The Powerpuff Girls",
    "summary": "<p>The superhero children have grown and are now disillusioned twenty-somethings who resent having lost their childhood to crime fighting. Will they agree to reunite now that the world needs them more than ever?</p>",
    "type": "show",
    "url": "https://www.tvmaze.com/shows/49892/the-powerpuff-girls"
  },
  {
    "name": "The Golden Girls",
    "summary": "<p><b>The Golden Girls</b> follows four South Florida seniors sharing a house, their dreams, and a whole lot of cheesecake. Bright, promiscuous, clueless, and hilarious, these lovely mismatched ladies form the perfect circle of friends.</p>",
    "type": "show",
    "url": "https://www.tvmaze.com/shows/722/the-golden-girls"
  },
  {
    "name": "Derry Girls",
    "summary": "<p>16-year-old Erin Quinn lives with her uncompromising mother, her long-suffering father and the fearsome 'Grandpa Joe', a man whose love for his daughters and granddaughters is surpassed only by his contempt for his son-in-law.</p><p>It's the early 90s, and Erin is used to seeing her country on the nightly news and speaking in acronyms (The IRA, The UDA, The RUC). This is a time of armed police in armoured Land Rovers and British Army check points. But it's also the time of Murder She Wrote, The Cranberries, MJ and Lisa Marie, Doc Martens, bomber jackets, The X Files, Nirvana and Wayne's World. And while The Troubles may hang over her home town, Erin has troubles of her own!</p>",
    "type": "show",
    "url": "https://www.tvmaze.com/shows/33320/derry-girls"
  }
]
```

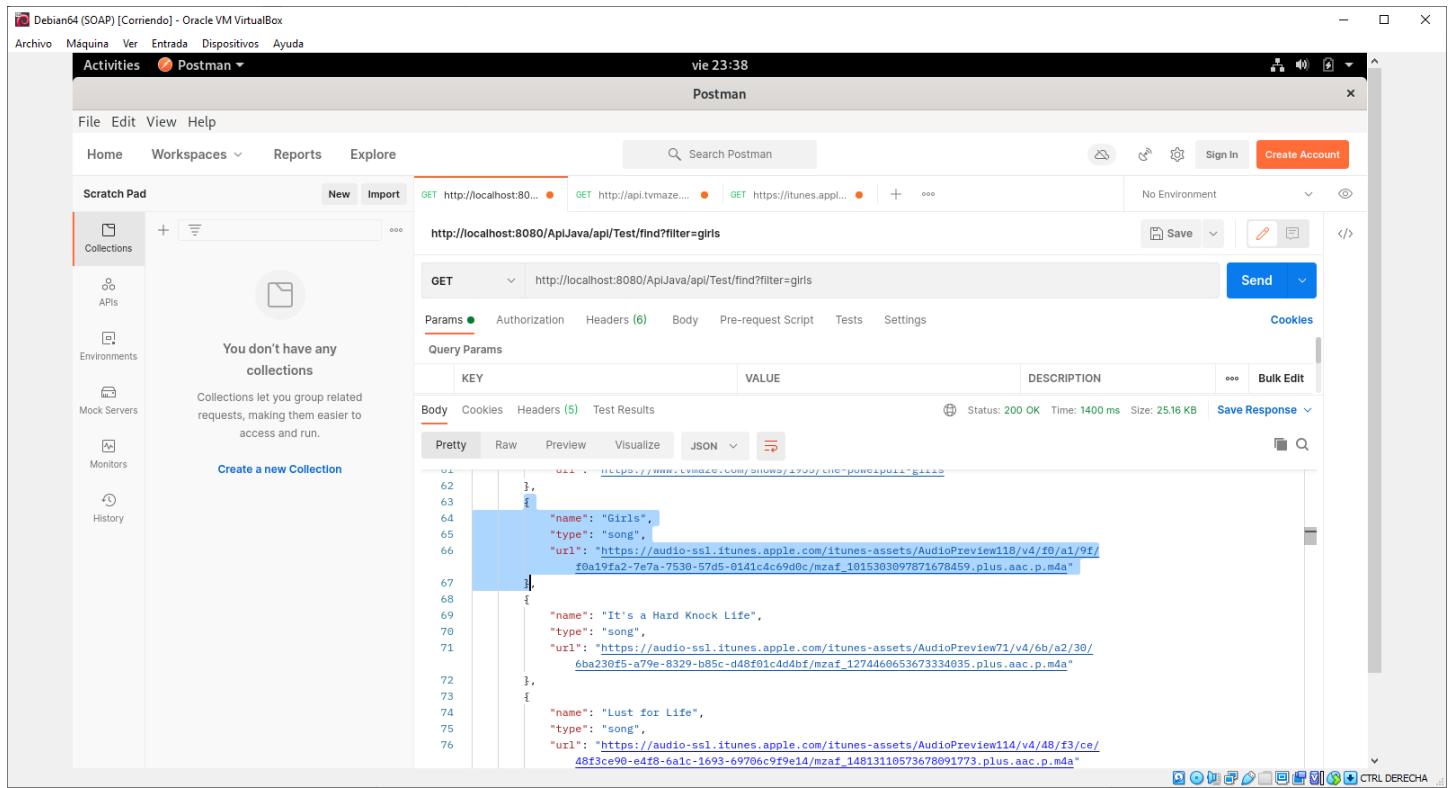
Así se devuelven los registros que corresponden a shows.



The screenshot shows a Postman interface with a GET request to `http://localhost:8080/ApiJava/api/Test/find?filter=girls`. The response body is identical to the one shown in the Firefox screenshot, listing five TV shows with their details. The Postman interface includes tabs for Params, Authorization, Headers, Body, Pre-request Script, Tests, and Settings, along with a detailed view of the JSON response.

API-TEST

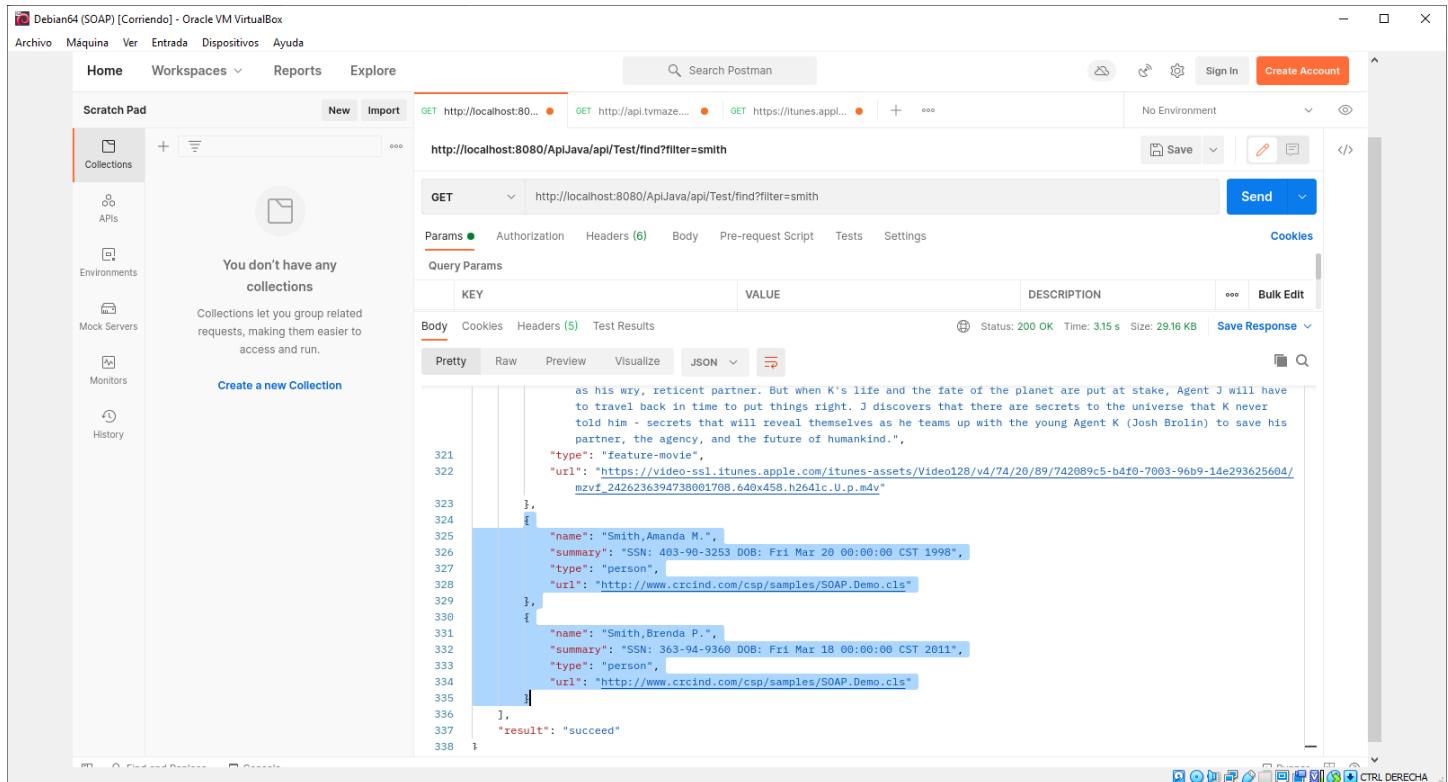
Así se devuelve los registros que corresponden a canciones y películas.



The screenshot shows the Postman interface with a GET request to `http://localhost:8080/ApiJava/api/Test/find?filter=girls`. The response body is a JSON array containing two items:

```
[{"name": "Girls", "type": "song", "url": "https://audio-ssl.itunes.apple.com/itunes-assets/AudioPreview118/v4/f0/a1/9f/6a19fa2-7e7a-7530-57d5-0141c4c69d0c/mzaf_101530309781678459.plus.aac.p.m4a"}, {"name": "It's a Hard Knock Life", "type": "song", "url": "https://audio-ssl.itunes.apple.com/itunes-assets/AudioPreview71/v4/6b/a2/30/6ba230f5-a79e-8329-b85c-d48f01c4d4bf/mzaf_1274460653673334035.plus.aac.p.m4a"}, {"name": "Lust for Life", "type": "song", "url": "https://audio-ssl.itunes.apple.com/itunes-assets/AudioPreview114/v4/48/f3/ce/48fce90-edfb-6a1c-1693-69786c979e14/mzaf_14813110573678091773.plus.aac.p.mda"}]
```

Así se devuelve los registros que corresponden a personas.



The screenshot shows the Postman interface with a GET request to `http://localhost:8080/ApiJava/api/Test/find?filter=smith`. The response body is a JSON object:

```
{ "name": "Smith,Amanda M.", "summary": "SSN: 463-90-3253 DOB: Fri Mar 20 00:00:00 CST 1998", "type": "person", "url": "http://www.crcind.com/csp/samples/SOAP_Demo.cls" }
```

Para implementarlo únicamente es necesario copiar el archivo `ApiJava.war` en la carpeta `webapps` en el directorio donde se haya instalado Tomcat

