



Máster Universitario En Investigación En Ingeniería De Software Y Sistemas Informáticos

Generación Automática de Código

Autor

César Hugo Bárzano Cruz



TRABAJO 1 DE EVALUACIÓN CONTINUA

—
2017/2018

Índice general

1. Introducción	7
2. Desarrollo de la Práctica	9
2.1. Tecnologías Utilizadas	9
2.2. Especificación de la Solución	9
2.3. Análisis de la solución	12
2.4. Pruebas	12
2.4.1. Pruebas de Funcionalidad	12
2.4.2. Pruebas de Error	20
3. Entrega	23
3.1. Generador	23
3.2. Makefile	23
3.3. Directorio software specification	23
3.4. Trabajo1_CesarHugoBarzanoCruz.pdf	23
3.5. Directorio DOC	24
3.6. Directorio input	24
3.7. Directorio Output	24
4. Anexo	27
4.1. generator.py	27
4.2. makefile	29

Índice de figuras

2.1. Test Strings	14
2.2. Test Numbers	14
2.3. Test Objects	14
2.4. Test Booleanos	15
2.5. Test Null	15
2.6. Test Arrays	16
2.7. Test XML to Json	20
2.8. Missing input file	21
2.9. Bad input format	21
2.10. Bad input json file	22

Capítulo 1

Introducción

El presente documento representa la memoria formal para evaluación continua de la segunda práctica de la asignatura Generación Automática de Código. Dicha práctica consiste en un generador de datos en formatos (CSV , HTML , XML, JSON) como salida a partir de un sistema de gestión de bases de datos (SGBD) relacional como entrada.

En los siguientes capítulos se detallará la solución propuesta, las tecnologías utilizadas, se analizarán las ventajas e inconvenientes de la solución propuesta, se explicarán los casos de uso o pruebas a los que se ha sometido el generador, se explicará como utilizar correctamente el generador así como las dependencias necesarias para su correcto uso.

Finalmente se analizarán las tareas y conocimientos adquiridos en esta práctica junto con un anexo de los documentos necesarios.

Capítulo 2

Desarrollo de la Práctica

2.1. Tecnologías Utilizadas

Como tecnología base, se ha decidido utilizar el lenguaje de programación interpretado Python[3], versión 2.7.15 debido a la sencillez y capacidad del mismo para proporcionar una solución.

Como se indicó en la introducción, la entrada del generador a impletar es uno o varios sistemas de gestión de bases de datos(SGBD). Los SGBDs que este generador soporta son los siguientes:

1. **mysql**
2. **postgres**
3. **sqlite3**

El sistema operativo base sobre el que se han realizado las pruebas es Ubuntu 18.04 LTS

Adicionalmente para la confección de esta memoria se ha utilizado LaTeX con el paquete de librerías texlive-full y el editor texmaker. Los siguientes enlaces muestran como instalar y utilizar correctamente LaTeX, han sido utilizados como referencia para el presente documento.

1. Instalar LaTeX
2. Usar LaTeX

2.2. Especificación de la Solución

Las posibles especificaciones e implementaciones para un generador como el propuesto por el enunciado pueden ser muy variadas. Dichas especificaciones pueden variar significativamente en función de las tecnologías utilizadas,

las aserciones o comprobaciones de robustez que se quieran realizar a las entradas, al procesamiento o a las salidas, los posibles casos que se quieran tener en cuenta así como el grado de modularidad que se quiera alcanzar de cara a futuras actualizaciones.

El generador utiliza a modo de configuración un fichero Json en el que vienen especificados los parametros necesarios para establecer la conexión con los distintos SGBD soportados. Un Ejemplo de entrada para este fichero de configuración es el siguiente:

Listing 2.1: Entrada Unitaria Configuración Generador

```
1 {  
2 ...  
3  
4 "mysql_1": {  
5     "user": "user",  
6     "password": "pass",  
7     "host": "host",  
8     "database": "database"  
9 },  
10 ...  
11 }
```

Las entradas usan como clave el identificador del tipo de SGBD, esto permite al generador diferenciar el tipo de conector a usar para poder extraer los datos, es decir, `mysql_n` establece la informacion necesaria para conectarse al SGBD número N basado en mysql. Esto permite configurar tantos SGBD basados en mysql como se quiera. El siguiente ejemplo muestra un ejemplo de configuración más completo en el que se incluyen todos los SGBD soportados por el generador:

Listing 2.2: Entrada Completa Configuración Generador

```
1 {  
2  
3     "mysql_1": {  
4         "user": "user",  
5         "password": "pass",  
6         "host": "host",  
7         "database": "database"  
8     },  
9     "mysql_2": {  
10         "user": "user2",  
11         "password": "pass2",  
12         "host": "host2",  
13         "database": "database2"  
14     },  
15     "postgres_1": {  
16         "user": "postgres",  
17         "password": "123456",  
18         "host": "host",  
19         "database": "database"  
20     },  
21     "sqlite3_1": {  
22         "db_path": "path/to/database/file"
```

```

23     }
24 }

```

Como podemos observar en el ejemplo anterior, el fichero de configuración para el generador nos permite definir tantos SGBD como necesitemos indicando siempre el indicador de tecnología: `mysql_N`, `postgres_N` o `sqlite3_N`.

En los casos de `mysql` y `postgres` es necesario establecer la configuración necesaria para la conexión indicando usuario, contraseña, host (url o ip) y base de datos a la que conectar. De manera adicional el generador soporta `sqlite3` pero debido a ser esta una base de datos para pruebas generalmente en local, la información necesaria es el path al fichero en disco que actúa como base de datos. Para este último caso es necesario tener en cuenta que dicho fichero tiene que tener los permisos adecuados (lectura al menos) para que el generador sea capaz de leer la información de dicha base de datos.

El generador cuenta con la siguiente ayuda con el objetivo de facilitar la comprensión de los argumentos que necesita para un uso nominal.

Listing 2.3: `python generadorP2.py -help`

```

1
2 Usage: python generador.py [options]
3
4 Options
5 -v, --version          Show the version of this script
6 -h, --help             Show this help.
7 -t <table_name>, --table <table_name>  Input database table
8 -w <"column = 'value'">, --where <"column = 'value'">  Optional Where
   Clause
9 -d, --debug            Debug Mode

```

Como podemos observar en la ayuda del generador, es necesario el flag `-t` o `--table` para indicar la tabla de la cual el generador va a extraer la información dentro de los distintos SGBD establecidos en la configuración.

De manera adicional, se puede añadir el flag `-w` o `--where` para filtrar utilizando la cláusula relacional `WHERE`.

Con el objetivo de facilitar la comprensión de que hace el generador en cada etapa de su ejecución, se ha incluido el flag `-d` o `--debug` el cual permite ver por la salida estándar que acción está realizando el generador.

Tras una ejecución, el generador creará en el directorio `OUTPUT` 4 ficheros por cada uno de los SGBD especificados con la siguiente convención de nombres:

1. `GAC_mysql_1_20180820_203859.csv`
2. `GAC_mysql_1_20180820_203859.html`
3. `GAC_mysql_1_20180820_203859.json`
4. `GAC_mysql_1_20180820_203859.xml`

2.3. Análisis de la solución

Se considera que la solución propuesta es valida, ya que cumple con lo solicitado, realiza las adecuadas comprobaciones para una correcta ejecución, genera los distintos formatos esperados y además se ha alcanzado de una manera sencilla, en un solo script python junto con los conectores necesarios para cada uno de los SGBD tratados. La modularidad de la solución propuesta favorece posibles actualizaciones del generador, incluyendo en el bucle de iteración principal el conector necesario para soportar nuevas bases de datos. El generador carga su configuración mediante variables de entorno (`GENERATOR_CONFIG`) lo que permite cambiar rapidamente el tipo de configuración a utilizar en cada ejecución.

La solución alcanzada no es completa debido a los conectores con las bases de datos, ya que como se ha explicado antes solo soporta 3 SGBD por lo que quizas esta seria su principal defecto.

2.4. Pruebas

Con el Objetivo de mostrar la potencia del generador, se van a realizar una serie de ejecuciones de prueba para mostrar todos los posibles escenarios del generador. Para ello es necesario disponer de los 3 tipos de SGBD soportadas por el generador con datos significativos que exportar. Para ello se incluye el fichero `init.py` cuyo objetivo es el de añadir entradas a las bases de datos instaladas localmente en mi equipo.

Para facilitar los distintos escenarios de pruebas a los que vamos a someter al generador, se va a utilizar el usuario `root` con password `root` para la base de datos `mysql`, una base de datos `postgres` cuyo usuario y contraseña es `postgres:123456` y como detallamos anteriormente una base de datos `sqlite3` representada por un fichero en disco.

Adicionalmente se va a mostrar ciertos casos de error tenidos en cuenta en la implementación de la solución, por lo que se propone dividir esta sección en las siguientes sub-secciones. Para automatizar este proceso se ha creado el `Makefile` que se adjunta en el anexo donde se especifica cada uno de los `testCases_N.sh` que serán ejecutados, tomando como entrada los ficheros alojados en el directorio `input` y dejando el resultado en el directorio `output`. En las capturas que muestran cada una de las pruebas se ve todo lo necesario para su ejecución.

2.4.1. Pruebas de Funcionalidad

Para mostrar la correcta funcionalidad del generador se va a comenzar exportando datos de cada uno de los SGBD de manera independiente. Tras esto, se ejemplificará el uso de la clausula `WHERE` para filtrar los datos a exportar y finalmente se mostrarán casos de pruebas mas complejos en los

que se atará con una sola ejecución a todos los SGBD soportados por el generador.

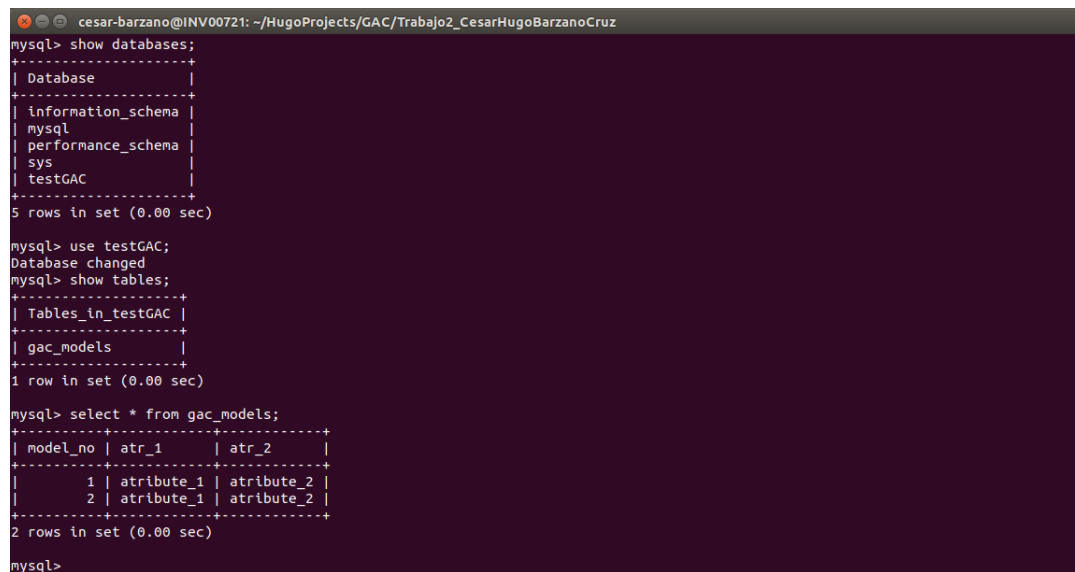
Test Case 1 - mysql

A continuación se muestra la configuración utilizada por el generador para el primer caso de prueba.

Listing 2.4: make testServlet

```
1 {
2     "mysql_1": {
3         "user": "root",
4         "password": "root",
5         "host": "localhost",
6         "database": "testGAC"
7     }
8 }
```

Utilizando la línea de órdenes para mysql, podemos observar cual es el estado interno del SGBD



```
cesar-barzano@INV00721: ~/HugoProjects/GAC/Trabajo2_CesarHugoBarzanoCruz
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| testGAC |
+-----+
5 rows in set (0.00 sec)

mysql> use testGAC;
Database changed
mysql> show tables;
+-----+
| Tables_in_testGAC |
+-----+
| gac_models |
+-----+
1 row in set (0.00 sec)

mysql> select * from gac_models;
+-----+-----+-----+
| model_no | atr_1 | atr_2 |
+-----+-----+-----+
| 1 | attribute_1 | attribute_2 |
| 2 | attribute_1 | attribute_2 |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Figura 2.1: Mysql status

Listing 2.5: make testServlet

```
1 1,attribute_1,attribute_2
2 2,attribute_1,attribute_2
```

Listing 2.6: make testServlet

```

1 <!DOCTYPE html>
2   <html>
3     <head>
4       <title>mysql_1</title>
5     </head>
6     <body>
7       <h1>mysql_1</h1><hr><br><table style="width:100%"><tr><td>2</td><
        td>atribute_1</td><td>atribute_2</td></tr></table></body></
        html>

```

Listing 2.7: make testServlet

```

1 [[1, "atribute_1", "atribute_2"], [2, "atribute_1", "atribute_2"]]

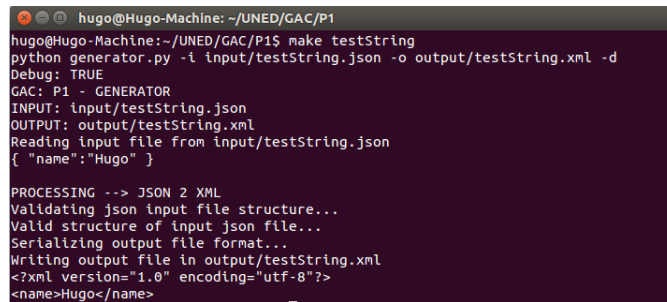
```

Listing 2.8: make testServlet

```

1 <data>&lt;data&gt;&lt;i&gt;&lt;i&gt;1&lt;/i&gt;&lt;i&gt;atribute_1&lt;
  ;/i&gt;&lt;i&gt;atribute_2&lt;/i&gt;&lt;i&gt;&lt;i&gt;&lt;i&gt;2&
  lt;/i&gt;&lt;i&gt;atribute_1&lt;/i&gt;&lt;i&gt;atribute_2&lt;/i&gt;
  &lt;/i&gt;&lt;/data&gt;</data>

```



```

hugo@Hugo-Machine: ~/UNED/GAC/P1
hugo@Hugo-Machine:~/UNED/GAC/P1$ make testString
python generator.py -i input/testString.json -o output/testString.xml -d
Debug: TRUE
GAC: P1 - GENERATOR
INPUT: input/testString.json
OUTPUT: output/testString.xml
Reading input file from input/testString.json
{ "name": "Hugo" }

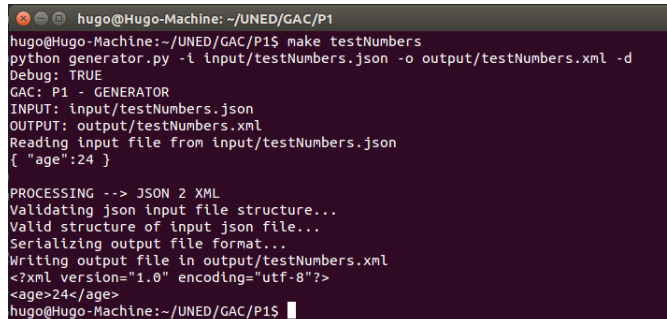
PROCESSING --> JSON 2 XML
Validating json input file structure...
Valid structure of input json file...
Serializing output file format...
Writing output file in output/testString.xml
<?xml version="1.0" encoding="utf-8"?>
<name>Hugo</name>

```

Figura 2.2: Test Strings

Numbers

Tipo de dato: Números



```

hugo@Hugo-Machine: ~/UNED/GAC/P1
hugo@Hugo-Machine:~/UNED/GAC/P1$ make testNumbers
python generator.py -i input/testNumbers.json -o output/testNumbers.xml -d
Debug: TRUE
GAC: P1 - GENERATOR
INPUT: input/testNumbers.json
OUTPUT: output/testNumbers.xml
Reading input file from input/testNumbers.json
{ "age": 24 }

PROCESSING --> JSON 2 XML
Validating json input file structure...
Valid structure of input json file...
Serializing output file format...
Writing output file in output/testNumbers.xml
<?xml version="1.0" encoding="utf-8"?>
<age>24</age>
hugo@Hugo-Machine:~/UNED/GAC/P1$

```

Figura 2.3: Test Numbers

Objects

Tipo de dato: Objetos Json

```
hugo@Hugo-Machine: ~/UNED/GAC/P1
hugo@Hugo-Machine:~/UNED/GAC/P1$ make testObjects
python generator.py -i input/testObjects.json -o output/testObjects.xml -d
Debug: TRUE
GAC: P1 - GENERATOR
INPUT: input/testObjects.json
OUTPUT: output/testObjects.xml
Reading input file from input/testObjects.json
{
  "alumno":{ "name":"Hugo", "age":24, "city":"Madrid" }
}

PROCESSING --> JSON 2 XML
Validating json input file structure...
Valid structure of input json file...
Serializing output file format...
Writing output file in output/testObjects.xml
<?xml version="1.0" encoding="utf-8"?>
<alumno>
  <city>Madrid</city>
  <age>24</age>
  <name>Hugo</name>
</alumno>
hugo@Hugo-Machine:~/UNED/GAC/P1$
```

Figura 2.4: Test Objects

Boolean

Tipo de dato: Booleanos

```
hugo@Hugo-Machine: ~/UNED/GAC/P1
hugo@Hugo-Machine:~/UNED/GAC/P1$ make testBoolean
python generator.py -i input/testBoolean.json -o output/testBoolean.xml -d
Debug: TRUE
GAC: P1 - GENERATOR
INPUT: input/testBoolean.json
OUTPUT: output/testBoolean.xml
Reading input file from input/testBoolean.json
{ "debug":true }

PROCESSING --> JSON 2 XML
Validating json input file structure...
Valid structure of input json file...
Serializing output file format...
Writing output file in output/testBoolean.xml
<?xml version="1.0" encoding="utf-8"?>
<debug>True</debug>
hugo@Hugo-Machine:~/UNED/GAC/P1$
```

Figura 2.5: Test Booleanos

Null

Tipo de dato: null

```

hugo@Hugo-Machine: ~/UNED/GAC/P1
hugo@Hugo-Machine:~/UNED/GAC/P1$ make testNull
python generator.py -i input/testNull.json -o output/testNull.xml -d
Debug: TRUE
GAC: P1 - GENERATOR
INPUT: input/testNull.json
OUTPUT: output/testNull.xml
Reading input file from input/testNull.json
{ "pareja":null }

PROCESSING --> JSON 2 XML
Validating json input file structure...
Valid structure of input json file...
Serializing output file format...
Writing output file in output/testNull.xml
<?xml version="1.0" encoding="utf-8"?>
<pareja></pareja>
hugo@Hugo-Machine:~/UNED/GAC/P1$

```

Figura 2.6: Test Null

Arrays

Tipo de dato: Listas

```

hugo@Hugo-Machine: ~/UNED/GAC/P1
hugo@Hugo-Machine:~/UNED/GAC/P1$ make testArrays
python generator.py -i input/testArrays.json -o output/testArrays.xml -d
Debug: TRUE
GAC: P1 - GENERATOR
INPUT: input/testArrays.json
OUTPUT: output/testArrays.xml
Reading input file from input/testArrays.json
{
  "planets": {
    "planet": [{"name": "Earth"}, {"name": "Jupiter"}, {"name": "Mars"}]
  }
}

PROCESSING --> JSON 2 XML
Validating json input file structure...
Valid structure of input json file...
Serializing output file format...
Writing output file in output/testArrays.xml
<?xml version="1.0" encoding="utf-8"?>
<planets>
  <planet>
    <name>Earth</name>
  </planet>
  <planet>
    <name>Jupiter</name>
  </planet>
  <planet>
    <name>Mars</name>
  </planet>
</planets>
hugo@Hugo-Machine:~/UNED/GAC/P1$

```

Figura 2.7: Test Arrays

Json complejos

La siguiente prueba muestra la capacidad del generador para trabajar con ficheros json complejos, como es por ejemplo el servlet de una aplicación web. Debido a la extensión de la salida de esta prueba, no se adjuntará captura, si no la salida estándar.

Listing 2.9: make testServlet

```

1 hugo@Hugo-Machine:~/UNED/GAC/P1$ make testServlet
2 python generator.py -i input/servlet.json -o output/output_testervlet
   .xml -d

```



```
3 Debug: TRUE
4 GAC: P1 - GENERATOR
5 INPUT: input/servlet.json
6 OUTPUT: output/output_testervlet.xml
7 Reading input file from input/servlet.json
8 {"web-app": {
9   "servlet": [
10    {
11      "servlet-name": "cofaxCDS",
12      "servlet-class": "org.cofax.cds.CDSServlet",
13      "init-param": {
14        "templateProcessorClass": "org.cofax.WysiwygTemplate",
15        "templateLoaderClass": "org.cofax.FilesTemplateLoader",
16        "templatePath": "templates",
17        "templateOverridePath": "",
18        "defaultListTemplate": "listTemplate.htm",
19        "defaultFileTemplate": "articleTemplate.htm",
20        "useJSP": false,
21        "jspListTemplate": "listTemplate.jsp",
22        "jspFileTemplate": "articleTemplate.jsp",
23        "cachePackageTagsTrack": 200,
24        "cachePackageTagsStore": 200,
25        "cachePackageTagsRefresh": 60,
26        "cacheTemplatesTrack": 100,
27        "cacheTemplatesStore": 50,
28        "cacheTemplatesRefresh": 15,
29        "cachePagesTrack": 200,
30        "cachePagesStore": 100,
31        "cachePagesRefresh": 10,
32        "cachePagesDirtyRead": 10,
33        "searchEngineListTemplate": "forSearchEnginesList.htm",
34        "searchEngineFileTemplate": "forSearchEngines.htm",
35        "searchEngineRobotsDb": "WEB-INF/robots.db",
36        "useDataStore": true,
37        "dataStoreClass": "org.cofax.SqlDataStore",
38        "redirectionClass": "org.cofax.SqlRedirection",
39        "dataStoreName": "cofax",
40        "dataStoreDriver": "com.microsoft.jdbc.sqlserver.
41          SQLServerDriver",
42        "dataStoreUrl": "jdbc:microsoft:sqlserver://LOCALHOST:1433;
43          DatabaseName=goon",
44        "dataStoreUser": "sa",
45        "dataStorePassword": "dataStoreTestQuery",
46        "dataStoreTestQuery": "SET NOCOUNT ON;select test='test';",
47        "dataStoreLogFile": "/usr/local/tomcat/logs/datastore.log",
48        "dataStoreInitConns": 10,
49        "dataStoreMaxConns": 100,
50        "dataStoreConnUsageLimit": 100,
51        "dataStoreLogLevel": "debug",
52        "maxUrlLength": 500}},
53    {
54      "servlet-name": "cofaxEmail",
55      "servlet-class": "org.cofax.cds.EmailServlet",
56      "init-param": {
57        "mailHost": "mail1",
58        "mailHostOverride": "mail2"}},
59    {
60      "servlet-name": "cofaxAdmin",
61      "servlet-class": "org.cofax.cds.AdminServlet"},
62    {
63      "servlet-name": "fileServlet",
```

```

63     "servlet-class": "org.cofax.cds.FileServlet"},
64     {
65         "servlet-name": "cofaxTools",
66         "servlet-class": "org.cofax.cms.CofaxToolsServlet",
67         "init-param": {
68             "templatePath": "toolstemplates/",
69             "log": 1,
70             "logLocation": "/usr/local/tomcat/logs/CofaxTools.log",
71             "logMaxSize": "",
72             "dataLog": 1,
73             "dataLogLocation": "/usr/local/tomcat/logs/dataLog.log",
74             "dataLogMaxSize": "",
75             "removePageCache": "/content/admin/remove?cache=pages&id=",
76             "removeTemplateCache": "/content/admin/remove?cache=templates&id=",
77             "fileTransferFolder": "/usr/local/tomcat/webapps/content/fileTransferFolder",
78             "lookInContext": 1,
79             "adminGroupID": 4,
80             "betaServer": true}}],
81     "servlet-mapping": {
82         "cofaxCDS": "/",
83         "cofaxEmail": "/cofaxutil/aemail/*",
84         "cofaxAdmin": "/admin/*",
85         "fileServlet": "/static/*",
86         "cofaxTools": "/tools/*"},
87
88     "taglib": {
89         "taglib-uri": "cofax.tld",
90         "taglib-location": "/WEB-INF/tlds/cofax.tld"}}}
91
92 PROCESSING --> JSON 2 XML
93 Validating json input file structure...
94 Valid structure of input json file...
95 Serializing output file format...
96 Writing output file in output/output_testervlet.xml
97 <?xml version="1.0" encoding="utf-8"?>
98 <web-app>
99     <servlet-mapping>
100         <cofaxTools>/tools/*</cofaxTools>
101         <cofaxCDS>/</cofaxCDS>
102         <fileServlet>/static/*</fileServlet>
103         <cofaxAdmin>/admin/*</cofaxAdmin>
104         <cofaxEmail>/cofaxutil/aemail/*</cofaxEmail>
105     </servlet-mapping>
106     <taglib>
107         <taglib-location>/WEB-INF/tlds/cofax.tld</taglib-location>
108         <taglib-uri>cofax.tld</taglib-uri>
109     </taglib>
110     <servlet>
111         <servlet-name>cofaxCDS</servlet-name>
112         <init-param>
113             <cachePagesStore>100</cachePagesStore>
114             <searchEngineListTemplate>forSearchEnginesList.htm</searchEngineListTemplate>
115             <maxUrlLength>500</maxUrlLength>
116             <dataStoreTestQuery>SET NOCOUNT ON;select test='test';</dataStoreTestQuery>
117             <defaultFileTemplate>articleTemplate.htm</defaultFileTemplate>
118             <dataStoreLogFile>/usr/local/tomcat/logs/

```

```

119         datastore.log</dataStoreLogFile>
120         <templateLoaderClass>org.cofax.
            FilesTemplateLoader</templateLoaderClass>
121         <dataStoreClass>org.cofax.SqlDataStore</
            dataStoreClass>
122         <templateOverridePath></templateOverridePath>
123         <cacheTemplatesStore>50</cacheTemplatesStore>
124         <dataStoreUrl>jdbc:microsoft:sqlserver://
            LOCALHOST:1433;DatabaseName=goon</
            dataStoreUrl>
125         <searchEngineFileTemplate>forSearchEngines.htm
            </searchEngineFileTemplate>
126         <cachePagesTrack>200</cachePagesTrack>
127         <cachePackageTagsStore>200</
            cachePackageTagsStore>
128         <dataStoreName>cofax</dataStoreName>
129         <dataStorePassword>dataStoreTestQuery</
            dataStorePassword>
130         <useJSP>False</useJSP>
131         <defaultListTemplate>listTemplate.htm</
            defaultListTemplate>
132         <dataStoreUser>sa</dataStoreUser>
133         <jspListTemplate>listTemplate.jsp</
            jspListTemplate>
134         <jspFileTemplate>articleTemplate.jsp</
            jspFileTemplate>
135         <dataStoreMaxConns>100</dataStoreMaxConns>
136         <cachePagesDirtyRead>10</cachePagesDirtyRead>
137         <cachePagesRefresh>10</cachePagesRefresh>
138         <cacheTemplatesTrack>100</cacheTemplatesTrack>
139         <dataStoreConnUsageLimit>100</
            dataStoreConnUsageLimit>
140         <redirectionClass>org.cofax.SqlRedirection</
            redirectionClass>
141         <searchEngineRobotsDb>WEB-INF/robots.db</
            searchEngineRobotsDb>
142         <templateProcessorClass>org.cofax.
            WysiwygTemplate</templateProcessorClass>
143         <cachePackageTagsRefresh>60</
            cachePackageTagsRefresh>
144         <templatePath>templates</templatePath>
145         <useDataStore>True</useDataStore>
146         <cacheTemplatesRefresh>15</
            cacheTemplatesRefresh>
147         <dataStoreDriver>com.microsoft.jdbc.sqlserver.
            SQLServerDriver</dataStoreDriver>
148         <cachePackageTagsTrack>200</
            cachePackageTagsTrack>
149         <dataStoreLogLevel>debug</dataStoreLogLevel>
150         <dataStoreInitConns>10</dataStoreInitConns>
151     </init-param>
152     <servlet-class>org.cofax.cds.CDSServlet</servlet-class>
153 </servlet>
154 <servlet>
155     <servlet-name>cofaxEmail</servlet-name>
156     <init-param>
157         <mailHostOverride>mail2</mailHostOverride>
158         <mailHost>mail1</mailHost>
159     </init-param>
    <servlet-class>org.cofax.cds.EmailServlet</servlet-
        class>

```

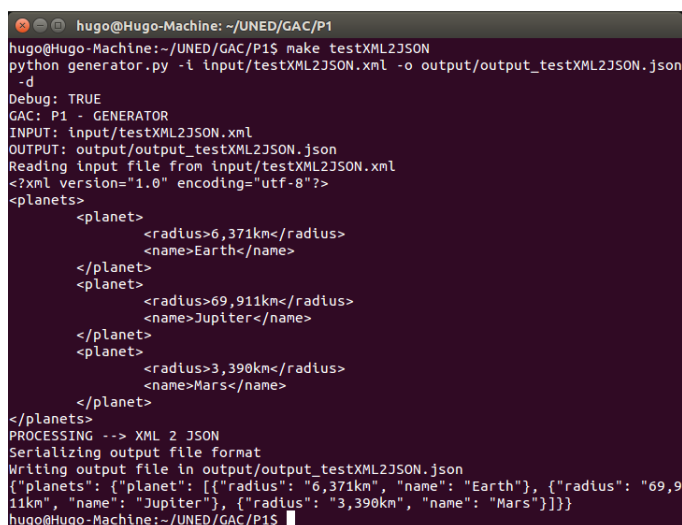
```

160         </servlet>
161         <servlet>
162             <servlet-name>cofaxAdmin</servlet-name>
163             <servlet-class>org.cofax.cds.AdminServlet</servlet-
                class>
164         </servlet>
165         <servlet>
166             <servlet-name>fileServlet</servlet-name>
167             <servlet-class>org.cofax.cds.FileServlet</servlet-
                class>
168         </servlet>
169         <servlet>
170             <servlet-name>cofaxTools</servlet-name>
171             <init-param>
172                 <logLocation>/usr/local/tomcat/logs/CofaxTools
                    .log</logLocation>
173                 <fileTransferFolder>/usr/local/tomcat/webapps/
                    content/fileTransferFolder</
                    fileTransferFolder>
174                 <log>1</log>
175                 <dataLog>1</dataLog>
176                 <dataLogLocation>/usr/local/tomcat/logs/
                    dataLog.log</dataLogLocation>
177                 <adminGroupID>4</adminGroupID>
178                 <lookInContext>1</lookInContext>
179                 <removePageCache>/content/admin/remove?cache=
                    pages&id=</removePageCache>
180                 <removeTemplateCache>/content/admin/remove?
                    cache=templates&id=</
                    removeTemplateCache>
181                 <logMaxSize></logMaxSize>
182                 <dataLogMaxSize></dataLogMaxSize>
183                 <betaServer>True</betaServer>
184                 <templatePath>toolstemplates/</templatePath>
185             </init-param>
186             <servlet-class>org.cofax.cms.CofaxToolsServlet</
                servlet-class>
187         </servlet>
188 </web-app>
189 hugo@Hugo-Machine: ~/UNED/GAC/P1$

```

XML to JSon

Como parte de la funcionalidad añadida, se va a realizar una única prueba, en la que el generador va a ingestar un fichero XML y va a generar un fichero JSON.



```
hugo@Hugo-Machine: ~/UNED/GAC/P1
hugo@Hugo-Machine:~/UNED/GAC/P1$ make testXML2JSON
python generator.py -i input/testXML2JSON.xml -o output/output_testXML2JSON.json
-d
Debug: TRUE
GAC: P1 - GENERATOR
INPUT: input/testXML2JSON.xml
OUTPUT: output/output_testXML2JSON.json
Reading input file from input/testXML2JSON.xml
<?xml version="1.0" encoding="utf-8"?>
<planets>
  <planet>
    <radius>6,371km</radius>
    <name>Earth</name>
  </planet>
  <planet>
    <radius>69,911km</radius>
    <name>Jupiter</name>
  </planet>
  <planet>
    <radius>3,390km</radius>
    <name>Mars</name>
  </planet>
</planets>
PROCESSING --> XML 2 JSON
Serializing output file format
Writing output file in output/output_testXML2JSON.json
{"planets": [{"planet": [{"radius": "6,371km", "name": "Earth"}, {"radius": "69,911km", "name": "Jupiter"}, {"radius": "3,390km", "name": "Mars"}]}}
hugo@Hugo-Machine:~/UNED/GAC/P1$
```

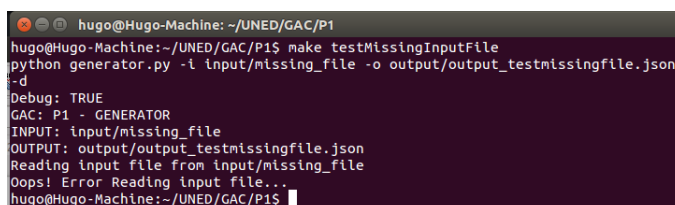
Figura 2.8: Test XML to Json

2.4.2. Pruebas de Error

Para realizar las pruebas de error, se ejercitarán los casos en los que las entradas no tengan el formato adecuado o la estructura del json de entrada no sea adecuada.

Entrada inexistente

Si el usuario intenta usar como entrada ficheros que no existen, el generador notificará el error y parará:



```
hugo@Hugo-Machine: ~/UNED/GAC/P1
hugo@Hugo-Machine:~/UNED/GAC/P1$ make testMissingInputFile
python generator.py -i input/missing_file -o output/output_testmissingfile.json
-d
Debug: TRUE
GAC: P1 - GENERATOR
INPUT: input/missing_file
OUTPUT: output/output_testmissingfile.json
Reading input file from input/missing_file
Oops! Error Reading input file...
hugo@Hugo-Machine:~/UNED/GAC/P1$
```

Figura 2.9: Missing input file

Input.format

Si el usuario del generador intenta usar como entrada ficheros cuya extensión no sea .json o .xml el generador notificará el error y parará:

```
hugo@Hugo-Machine: ~/UNED/GAC/P1
hugo@Hugo-Machine:~/UNED/GAC/P1$ make testBadInputFormat
python generator.py -i input/input.format -o output/output_testBadInputFormat.js
on -d
Debug: TRUE
GAC: P1 - GENERATOR
INPUT: input/input.format
OUTPUT: output/output_testBadInputFormat.json
Reading input file from input/input.format

Invalid input file format. It should be input.json or input.xml
hugo@Hugo-Machine:~/UNED/GAC/P1$
```

Figura 2.10: Bad input format

Estructura Json

Si el usuario del generador intenta usar como entrada un fichero json cuya estructura no sea la adecuada, el generador notificará el error y parará ya que realiza un chequeo contra esquema solo para los json de entrada.

```
hugo@Hugo-Machine: ~/UNED/GAC/P1
hugo@Hugo-Machine:~/UNED/GAC/P1$ make testBadJson
python generator.py -i input/badInput.json -o output/output_testBADInput.xml -d
Debug: TRUE
GAC: P1 - GENERATOR
INPUT: input/badInput.json
OUTPUT: output/output_testBADInput.xml
Reading input file from input/badInput.json
{
  "planets": {
    "planet":
    {
      "name": "Earth",
      "radius": "6,371km"
    },
    {
      "name": "Jupiter",
      "radius": "69,911km"
    },
    {
      "name": "Mars",
      "radius": "3,390km"
    }
  }
}

PROCESSING --> JSON 2 XML
Validating json input file structure...
Oops! Invalid structure of input json file...
hugo@Hugo-Machine:~/UNED/GAC/P1$
```

Figura 2.11: Bad input json file

Capítulo 3

Entrega

En este capítulo se detallan cada uno de los ficheros/directorios que forman parte de la entrega.

3.1. Generador

El generador propuesto por el enunciado se compone de un único fichero denominado `generator.py` el cual se encuentra alojado en la raíz del fichero comprimido que se envía. El generador ha sido desarrollado sobre Ubuntu14 y hace uso de la versión python 2.7.6 la cual incluye todas las dependencias necesarias. Si alguna de ellas causará error, podría ser instalada fácilmente descargando el paquete de las URLs indicadas en la bibliografía y ejecutando `sudo pip install nombre_paquete`

3.2. Makefile

El fichero "makefile" establece las distintas ejecuciones de pruebas para el generador. También dispone de la entrada "make doc" para generar la especificación software mediante pydoc. Alojado en la raíz del fichero comprimido para la entrega.

3.3. Directorio software specification

Contiene la especificación del generador en formato HTML. Dicha especificación ha sido generada utilizando la librería pydoc.

3.4. Trabajo1_CesarHugoBarzanoCruz.pdf

Memoria de la práctica, referencia a este documento en si mismo, alojado en el directorio raíz de la entrega.

3.5. Directorio DOC

Directorio donde se almacenan todos los fuentes usados para generar esta documentación utilizando LaTeX. Incluye tambien las imagenes usadas en la memoria.

3.6. Directorio input

Directorio donde se almacenan todos los ficheros usados como entradas en la sección de pruebas

3.7. Directorio Output

Directorio donde se almacenan todos los ficheros resultados de la ejecución de la pruebas.

Bibliografía

- [1] XML, *XML Format* <https://www.w3.org/XML/>
- [2] JSON, *JSON Format* <https://www.json.org/>
- [3] PYTHON, *Python 2.7.6 language* <https://www.python.org/download/releases/2.7.6/>
- [4] IMPORT JSON, *JSON encoder and decoder* <https://docs.python.org/2/library/json.html>
- [5] IMPORT XMLTODIC, *Makes working with XML feel like you are working with JSON* <https://pypi.python.org/pypi/xmltodict>
- [6] IMPORT SYS, *System-specific parameters and functions* <https://docs.python.org/2/library/sys.html>
- [7] IMPORT OS, *Miscellaneous operating system interfaces* <https://docs.python.org/2/library/os.html>
- [8] IMPORT GETOPT, *C-style parser for command line options* <https://docs.python.org/2/library/getopt.html>

Capítulo 4

Anexo

4.1. generator.py

Listing 4.1: generator.py

```
1  """XML generator from file.json"""
2
3
4  __author__ = 'Hugo Barzano'
5  __date__ = '2017/2018'
6  __version__ = 'v1.0'
7  __credits__ = 'GAC'
8  __file__ = 'generator.py'
9
10 import json
11 import xmldict
12 import sys
13 import os
14 import getopt
15
16 def readInput(input_file):
17     """Function to read input files.
18         :param input_file: input file path"""
19     print "Reading input file from "+input_file
20     try:
21         with open(input_file, 'r') as f:
22             input_file_as_string = f.read()
23         return input_file_as_string
24     except EnvironmentError:
25         print("Oops! Error Reading input file...")
26         exit()
27
28 def writeOutput(output_file,output_string):
29     """Function to write output files.
30         :param output_file: output file path
31         :param output_string: string to write in the output file"""
32     print "Writing output file in "+output_file
33     try:
34         with open(output_file, 'w') as f:
35             f.write(output_string)
36     except EnvironmentError:
37         print("Oops! Error writing output file...")
```

```

38         exit()
39
40 def validateJson(input_string):
41     """Function to validate json inputs.
42     :param input_string: json as string to validate"""
43     try:
44         validate_input=json.loads(input_string)
45         print "Valid structure of input json file..."
46     except ValueError:
47         print("Oops! Invalid structure of input json file...")
48         exit()
49
50 def parseInputJsonToXML(input_string):
51     """Function to parse json files as input to XML string.
52     :param input_string: input data file as string """
53     xmlString = xmltodict.unparse(json.loads(input_string), pretty=
54         True)
55     return xmlString
56
57 def parseInputXMLToJson(input_string):
58     """Function to parse xml files as input to json string.
59     :param input_string: input data file as string """
60     jsonString = json.dumps(xmltodict.parse(input_string))
61     return jsonString
62
63 def usage():
64     """Method to display generator usage. """
65     print """
66 Usage: python generator.py [options]
67
68 Options
69 -v, --version          Show the version of this script
70 -h, --help             Show this help.
71 -i <path>, --input <path>    Input file
72 -o <path>, --output <path>    Output file
73 -d, --debug            Debug Mode
74 """
75
76 def version():
77     """Function to display software version"""
78     print "version 1.0"
79
80 def main(argv):
81     """Main method to execute the generator.
82     :param argv: value from 0 to n-1 to ident"""
83     input_file=None
84     output_file=None
85     debug=False
86
87     # process arguments
88     try:
89         opts, args = getopt.getopt(argv, "hvi:o:d", ["help", "version",
90             "input=", "output=", "debug"])
91     except getopt.GetoptError as err:
92         print str(err)
93         usage()
94         sys.exit(2)
95     for opt, arg in opts:
96         if opt in ("-h", "--help"):
97             usage()
98             sys.exit()
99         elif opt in ("-v", "--version"):

```

```

98         version()
99         sys.exit()
100     elif opt in ("-i", "--input"):
101         input_file = arg
102     elif opt in ("-o", "--output"):
103         output_file=arg
104     elif opt in ("-d", "--debug"):
105         debug = True
106         print "Debug: TRUE"
107     else:
108         usage()
109         sys.exit()
110
111     print "GAC: P1 - GENERATOR"
112     if debug:
113         print "INPUT: "+ input_file
114         print "OUTPUT: "+ output_file
115
116
117
118     input_as_string=readInput(input_file)
119     if debug: print input_as_string
120     if input_file.split(".")[1]=="json":
121         print "PROCESSING --> JSON 2 XML "
122         if debug: print "Validating json input file structure..."
123         validateJson(input_as_string)
124         if debug: print "Serializing output file format..."
125         if not ".xml" in output_file: output_file=output_file+".xml"
126         output_as_string=parseInputJsonToXML(input_as_string)
127     elif input_file.split(".")[1]=="xml":
128         print "PROCESSING --> XML 2 JSON"
129         if debug: print "Serializing output file format"
130         if not ".json" in output_file: output_file=output_file+".json"
131         output_as_string=parseInputXMLToJson(input_as_string)
132     else:
133         print "Invalid input file format. It should be input.json or
134             input.xml "
135         exit()
136     writeOutput(output_file,output_as_string)
137     if debug:print output_as_string
138
139 #
140 # -----
141 #
142
143 if __name__ == "__main__":
144     if len(sys.argv) > 1:
145         main(sys.argv[1:])
146     else:
147         usage()
148         sys.exit()

```

4.2. makefile

Listing 4.2: makefile

```
1 #Makefile
2
3 doc:
4     rm -r software_specification && mkdir software_specification
5     && pydoc -w ./ && mv ./*.html software_specification/
6
7 testString:
8     python generator.py -i input/testString.json -o output/
9     testString.xml -d
10
11 testNumbers:
12     python generator.py -i input/testNumbers.json -o output/
13     testNumbers.xml -d
14
15 testObjects:
16     python generator.py -i input/testObjects.json -o output/
17     testObjects.xml -d
18
19 testBoolean:
20     python generator.py -i input/testBoolean.json -o
21     output/testBoolean.xml -d
22
23 testNull:
24     python generator.py -i input/testNull.json -o output/
25     testNull.xml -d
26
27 testArrays:
28     python generator.py -i input/testArrays.json -o output/
29     testArrays.xml -d
30
31 testServlet:
32     python generator.py -i input/servlet.json -o output/
33     output_testervlet.xml -d
34
35 testXML2JSON:
36     python generator.py -i input/testXML2JSON.xml -o
37     output/output_testXML2JSON.json -d
38
39 testBadInputFormat:
40     python generator.py -i input/input.format -o output/
41     output_testBadInputFormat.xml -d
42
43 testMissingInputFile:
44     python generator.py -i input/missing_file -o output/
45     output_testmissingfile.xml -d
46
47 testBadJson:
48     python generator.py -i input/badInput.json -o output/
49     output_testBADInput.xml -d
```