



**EDUCACIÓN**  
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO  
NACIONAL DE MÉXICO®



INSTITUTO TECNOLÓGICO DE MORELIA  
*"José María Morelos y Pavón"*

# **INSTITUTO TECNOLÓGICO DE MORELIA**

## **DIVISIÓN DE ESTUDIOS PROFESIONALES**

### **Lenguajes y Autómatas 2**

### **Proyecto Unidad 2**

### **Generador de código semántico**

PRESENTA:

**Daniel Avila Vergara**  
**Elliot Rendon León**  
**Luis Ignacio Manríquez Infante**

ASESOR:

**J. GUADALUPE RAMOS DIAZ**

**MORELIA, MICHOACÁN**

## ÍNDICE

Generador de código semántico .....	1
Desarrollo de la gramática biblioteca:.....	1
Desarrollo de las gramáticas Individuales: .....	3
Usuario .....	3
Generación RDF Usuario: .....	3
Libros .....	5
Generación RDF Libro:.....	6
Préstamo .....	9
Generación RDF Libro:.....	9
Gramática RDF.....	11
RDF Validator .....	14
Integración en IDE .....	15

## Generador de código semántico

Para comenzar con la explicación, tenemos que empezar por definir de manera corta que es un generador de código semántico, según la biblioteca de datos de IBM podemos definir al generador de código semántico una herramienta o tecnología que ayuda a convertir código no semántico o no estructurado en código HTML semántico, es decir, en código que utiliza etiquetas y elementos HTML para describir el significado y la estructura del contenido de una página web, con esta definición podemos comenzar con el desarrollo de la actividad.

### *Desarrollo de la gramática biblioteca:*

Como primer paso analizaremos por completo el funcionamiento de la gramática principal, la cual destaca por tener las 3 generaciones de código RDF en el mismo archivo, el código de la gramática es el siguiente:

```
grammar biblioteca;

inicio: seccion+;

// Por seccion debe de procesar cada tipo de instancia (libro, usuario, prestamo)
seccion
: 'Libros' registrosLibros? WS?
| 'Usuarios' registrosUsuarios? WS?
| 'Prestamos' registrosPrestamos? WS?
;

// Registros de Libros
registrosLibros: (libro)+;

libro
: id=INT COMMA
  nombre=STRING COMMA
  editorial=STRING COMMA
  autor=STRING COMMA
  genero=STRING COMMA
  paisAutor=STRING COMMA
  numeroPaginas=INT COMMA
  anioEdicion=DATE COMMA
  precio=DOUBLE
{
    // Generar RDF para el libro
    System.out.println("<!-- Instancia libro -->");
    System.out.println("<rdf:Description rdf:about=\"#"codigoLibro " + $id.text+"\">");
    System.out.println("  <rdf:type rdf:resource=\"#"Libro\" />");
    System.out.println("  <biblioteca:nombreLibro>"+$nombre.text+"</biblioteca:nombreLibro>");
    System.out.println("  <biblioteca:editorial>"+$editorial.text+"</biblioteca:editorial>");
    System.out.println("  <biblioteca:autor>"+$autor.text+"</biblioteca:autor>");
    System.out.println("  <biblioteca:genero>"+$genero.text+"</biblioteca:genero>");
    System.out.println("  <biblioteca:paisAutor>"+$paisAutor.text+"</biblioteca:paisAutor>");
    System.out.println("  <biblioteca:numeroPaginas>"+$numeroPaginas.text+"</biblioteca:numeroPaginas>");
    System.out.println("  <biblioteca:anioEdicion>"+$anioEdicion.text+"</biblioteca:anioEdicion>");
    System.out.println("  <biblioteca:precioLibro>"+$precio.text+"</biblioteca:precioLibro>");
    System.out.println("</rdf:Description>\n");
};

// Registros de Usuarios
registrosUsuarios: (usuario)+;
```

```

// Registros de Usuarios
registrosUsuarios: (usuario)+;

usuario:
    id=INT COMMA
    nombre=STRING COMMA
    apellido=STRING COMMA
    ine=INT COMMA
    domicilio=STRING COMMA
    estado=STRING COMMA
    municipio=STRING COMMA
    nacimiento=DATE
{
    // Generar RDF para el usuario
    System.out.println("<!-- Instancia usuario -->");
    System.out.println("<rdf:Description rdf:about=\"#" + $id.text + "\">");
    System.out.println("  <rdf:type rdf:resource=\"#Usuario\" />");
    System.out.println("  <biblioteca:nombre>"+$nombre.text+"</biblioteca:nombre>");
    System.out.println("  <biblioteca:apellidos>"+$apellido.text+"</biblioteca:apellidos>");
    System.out.println("  <biblioteca:noIdentificacion>"+$ine.text+"</biblioteca:noIdentificacion>");
    System.out.println("  <biblioteca:domicilio>"+$domicilio.text+"</biblioteca:domicilio>");
    System.out.println("  <biblioteca:estado>"+$estado.text+"</biblioteca:estado>");
    System.out.println("  <biblioteca:municipio>"+$municipio.text+"</biblioteca:municipio>");
    System.out.println("  <biblioteca:nacimiento>"+$nacimiento.text+"</biblioteca:nacimiento>");
    System.out.println("</rdf:Description>\n");
};

// Registros de Préstamos
registrosPrestamos: (prestamo)+;

prestamo:
    numeroPedido=INT COMMA
    codigoLibro=INT COMMA
    codigoUsuario=INT COMMA
    fechaSalida=DATE COMMA
    fechaMaxima=DATE COMMA
    fechaDevolucion=DATE
{
    // Generar RDF para el préstamo
    System.out.println("<!-- Instancia préstamo -->");
    System.out.println("<rdf:Description rdf:about=\"#numeroPedido "+$codigoLibro.text+"\">");
    System.out.println("  <rdf:type rdf:resource=\"#Prestamo\" />");
    System.out.println("  <biblioteca:codigoLibro>"+$codigoLibro.text+"</biblioteca:codigoLibro>");
    System.out.println("  <biblioteca:codigoUsuario>"+$codigoUsuario.text+"</biblioteca:codigoUsuario>");
    System.out.println("  <biblioteca:fechaSalida>"+$fechaSalida.text+"</biblioteca:fechaSalida>");
    System.out.println("  <biblioteca:fechaMaxima>"+$fechaMaxima.text+"</biblioteca:fechaMaxima>");
    System.out.println("  <biblioteca:fechaDevolucion>"+$fechaDevolucion.text+"</biblioteca:fechaDevolucion>");
    System.out.println("</rdf:Description>\n");
};

INT: ('0' .. '9')+;
DOUBLE: INT '.' INT;
STRING: ('a' .. 'z' | 'A' .. 'Z' | '_' | ('a' .. 'z' | 'A' .. 'Z' | '0' .. '9' | '_' | ' '))+;
DATE: INT '/' INT '/' INT;
COMMA: ',';
WS: (' ' | '\n' | '\t' | '\r')+ {$channel=HIDDEN};

```

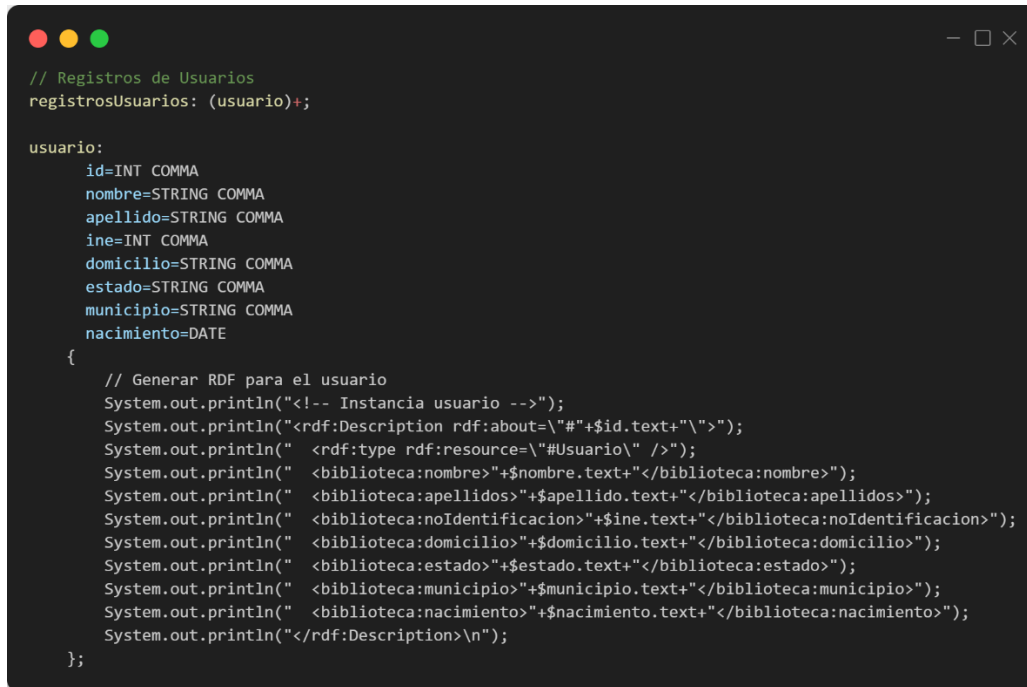
¿Como podríamos describir esta parte?, bueno primero tenemos que tener conciencia de que la gramática está diseñada para convertir registros textuales sobre libros, usuarios y préstamos en representaciones RDF que pueden ser fácilmente procesadas y consumidas en la web semántica, analizando los datos que llegan de entrada pues cuando se detecta un registro válido, se genera un bloque RDF que describe ese registro.

En donde cada registro de libros, usuarios o préstamos se traduce a un bloque RDF en formato XML con etiquetas específicas para cada tipo de dato (Cosa que veremos analizaremos más adelante).

### *Desarrollo de las gramáticas Individuales:*

#### *Usuario*

Se definió una estructura para procesar información de usuarios y generar datos en formato RDF (Figura 1).



```
// Registros de Usuarios
registrosUsuarios: (usuario)+;

usuario:
    id=INT COMMA
    nombre=STRING COMMA
    apellido=STRING COMMA
    ine=INT COMMA
    domicilio=STRING COMMA
    estado=STRING COMMA
    municipio=STRING COMMA
    nacimiento=DATE
{
    // Generar RDF para el usuario
    System.out.println("<!-- Instancia usuario -->");
    System.out.println("<rdf:Description rdf:about=\"#" + $id.text + "\">");
    System.out.println("    <rdf:type rdf:resource=\"#Usuario\" />");
    System.out.println("    <biblioteca:nombre>"+$nombre.text+"</biblioteca:nombre>");
    System.out.println("    <biblioteca:apellidos>"+$apellido.text+"</biblioteca:apellidos>");
    System.out.println("    <biblioteca:noIdentificacion>"+$ine.text+"</biblioteca:noIdentificacion>");
    System.out.println("    <biblioteca:domicilio>"+$domicilio.text+"</biblioteca:domicilio>");
    System.out.println("    <biblioteca:estado>"+$estado.text+"</biblioteca:estado>");
    System.out.println("    <biblioteca:municipio>"+$municipio.text+"</biblioteca:municipio>");
    System.out.println("    <biblioteca:nacimiento>"+$nacimiento.text+"</biblioteca:nacimiento>");
    System.out.println("</rdf:Description>\n");
};
```

Figura 1

Como primera parte de la explicación del código, tenemos que estructurar la información de un usuario, en este caso usamos la estructura representada por un “ID” (el identificador único del usuario), un “NOMBRE” (El nombre del usuario), un “APELLIDO” (apellido del usuario) , el “INE” (representación de la identificación oficial del usuario), el “Domicilio”, “ESTADO” y “DOMICILIO” (representación del domicilio del usuario) y el “NACIMIENTO” (que representa la fecha de nacimiento del usuario), estos datos en general son la estructura de como tendrían que llegar los registros para poder ser convertidos a formato RDF como veremos a continuación.

#### *Generación RDF Usuario:*

El bloque llamado “Generación RDF” es el encargado de generar el código RDF a partir de los valores capturados, esto sucede cada vez que se reconoce una instancia de usuario, permitiendo automatizar la generación de los formatos RDF de cada uno de los usuarios registrados (Figura 2).

```

101,Juan,Perez,8765432101234,Calle Falsa 123,CDMX,Ciudad de Mexico,15/02/1990
102,Ana,Gomez,2345678901234,Avenida Siempre Viva 742,CDMX,Ciudad de Mexico,25/06/1985

^Z
<rdf:Description rdf:about="#101">
  <rdf:type rdf:resource="#Usuario" />
  <biblioteca:codigoUsuario>101</biblioteca:codigoUsuario>
  <biblioteca:nombre>Juan</biblioteca:nombre>
  <biblioteca:apellidos>Perez</biblioteca:apellidos>
  <biblioteca:noIdentificacion>8765432101234</biblioteca:noIdentificacion>
  <biblioteca:domicilio>Calle Falsa 123</biblioteca:domicilio>
  <biblioteca:estado>CDMX</biblioteca:estado>
  <biblioteca:municipio>Ciudad de Mexico</biblioteca:municipio>
  <biblioteca:nacimiento>15/02/1990</biblioteca:nacimiento>
</rdf:Description>

<rdf:Description rdf:about="#102">
  <rdf:type rdf:resource="#Usuario" />
  <biblioteca:codigoUsuario>102</biblioteca:codigoUsuario>
  <biblioteca:nombre>Ana</biblioteca:nombre>
  <biblioteca:apellidos>Gomez</biblioteca:apellidos>
  <biblioteca:noIdentificacion>2345678901234</biblioteca:noIdentificacion>
  <biblioteca:domicilio>Avenida Siempre Viva 742</biblioteca:domicilio>
  <biblioteca:estado>CDMX</biblioteca:estado>
  <biblioteca:municipio>Ciudad de Mexico</biblioteca:municipio>
  <biblioteca:nacimiento>25/06/1985</biblioteca:nacimiento>
</rdf:Description>

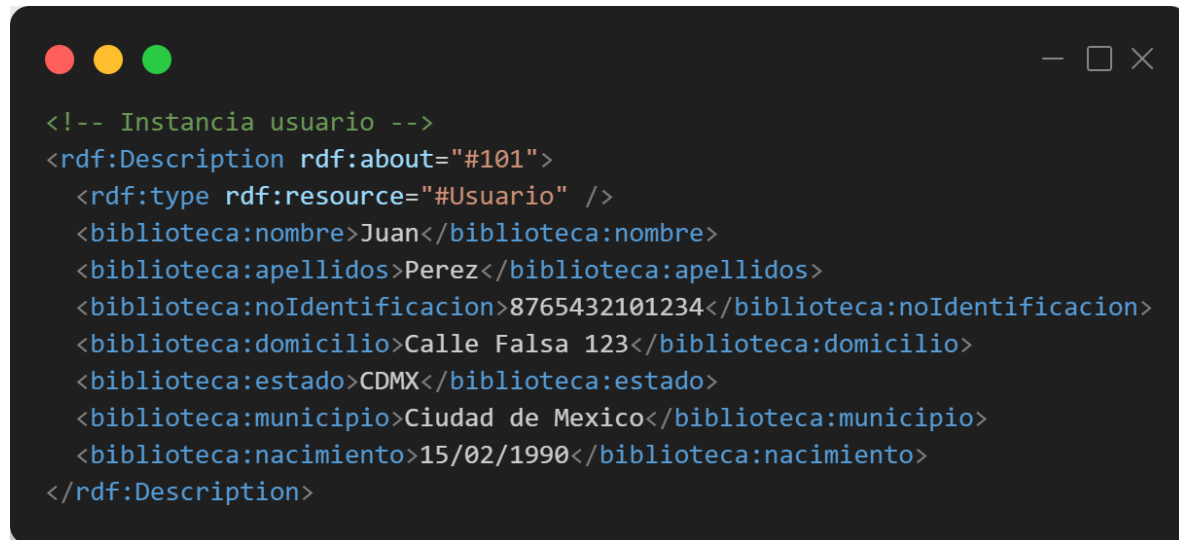
```

Figura 2

En este caso, ¿Que podemos destacar de la ejecución de este código?, como primera parte notamos que tal como se explicó en la figura anterior, la estructura de la instancia del usuario generara el código RDF del mismo, un ejemplo claro de esto es la siguiente instancia:

**101, Juan, Perez,8765432101234, Calle Falsa 123, CDMX, Ciudad de Mexico,15/02/1990**

A partir de la cual se genera el código RDF correspondiente (Figura 3)

A screenshot of a code editor window with a dark background. The code is written in XML-like syntax for RDF. It starts with a comment: <!-- Instancia usuario -->. Then it opens an <rdf:Description rdf:about="#101"> tag. Inside, it defines the type as <rdf:type rdf:resource="#Usuario" />. Then it lists several properties: <biblioteca:nombre>Juan</biblioteca:nombre>, <biblioteca:apellidos>Perez</biblioteca:apellidos>, <biblioteca:noIdentificacion>8765432101234</biblioteca:noIdentificacion>, <biblioteca:domicilio>Calle Falsa 123</biblioteca:domicilio>, <biblioteca:estado>CDMX</biblioteca:estado>, <biblioteca:municipio>Ciudad de Mexico</biblioteca:municipio>, and <biblioteca:nacimiento>15/02/1990</biblioteca:nacimiento>. Finally, it closes the </rdf:Description> tag. The code is color-coded: comments are green, tags are blue, and values are white. The editor has standard window controls (red, yellow, green buttons and minus, maximize, close icons) in the top right corner.

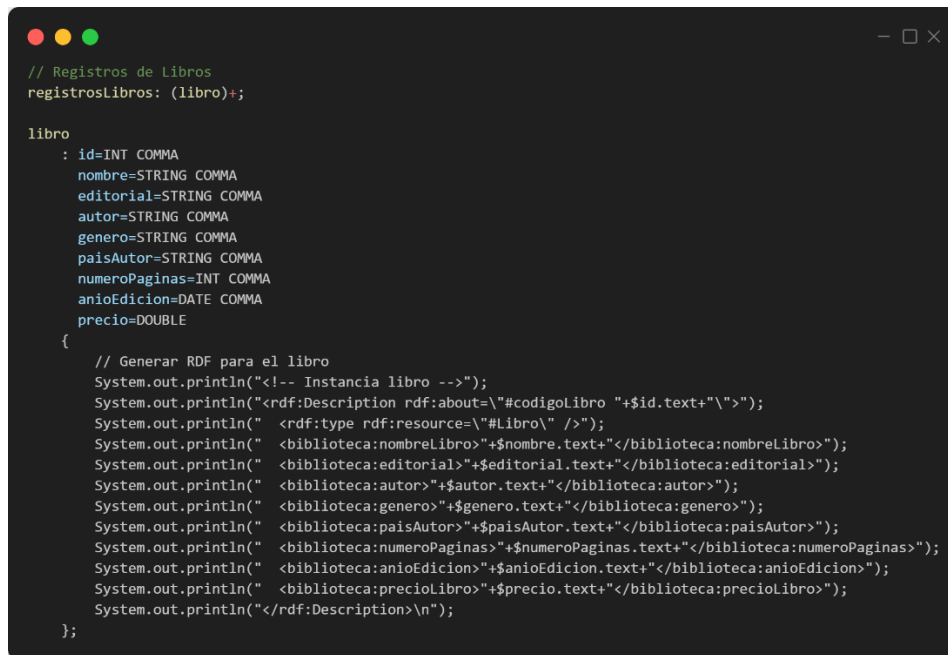
```
<!-- Instancia usuario -->
<rdf:Description rdf:about="#101">
  <rdf:type rdf:resource="#Usuario" />
  <biblioteca:nombre>Juan</biblioteca:nombre>
  <biblioteca:apellidos>Perez</biblioteca:apellidos>
  <biblioteca:noIdentificacion>8765432101234</biblioteca:noIdentificacion>
  <biblioteca:domicilio>Calle Falsa 123</biblioteca:domicilio>
  <biblioteca:estado>CDMX</biblioteca:estado>
  <biblioteca:municipio>Ciudad de Mexico</biblioteca:municipio>
  <biblioteca:nacimiento>15/02/1990</biblioteca:nacimiento>
</rdf:Description>
```

Figura 3

Al igual que en el resto de las gramáticas podemos decir que validar que cumplan con un formato específico, al cumplir con este requisito podemos transformar cada registro de usuario en una representación RDF.

### *Libros*

Se definió una estructura para procesar información de los libros para generar datos en formato RDF (Figura 4).



```
// Registros de Libros
registrosLibros: (libro)+;

libro
: id=INT COMMA
  nombre=STRING COMMA
  editorial=STRING COMMA
  autor=STRING COMMA
  genero=STRING COMMA
  paisAutor=STRING COMMA
  numeroPaginas=INT COMMA
  anioEdicion=DATE COMMA
  precio=DOUBLE
{
  // Generar RDF para el libro
  System.out.println("<!-- Instancia libro -->");
  System.out.println("<rdf:Description rdf:about=\"\#codigolibro "+$id.text+"\">");
  System.out.println("  <rdf:type rdf:resource=\"\#Libro\" />");
  System.out.println("  <biblioteca:nombreLibro>"+$nombre.text+"</biblioteca:nombreLibro>");
  System.out.println("  <biblioteca:editorial>"+$editorial.text+"</biblioteca:editorial>");
  System.out.println("  <biblioteca:autor>"+$autor.text+"</biblioteca:autor>");
  System.out.println("  <biblioteca:genero>"+$genero.text+"</biblioteca:genero>");
  System.out.println("  <biblioteca:paisAutor>"+$paisAutor.text+"</biblioteca:paisAutor>");
  System.out.println("  <biblioteca:numeroPaginas>"+$numeroPaginas.text+"</biblioteca:numeroPaginas>");
  System.out.println("  <biblioteca:anioEdicion>"+$anioEdicion.text+"</biblioteca:anioEdicion>");
  System.out.println("  <biblioteca:precioLibro>"+$precio.text+"</biblioteca:precioLibro>");
  System.out.println("</rdf:Description>\n");
};
```

Figura 4

En esta gramática podemos ver la organización de los campos como “ID” (Identificador único del libro), el “NOMBRE”, “EDITORIAL”, “AUTOR”, “GENERO”, “PAIS\_AUTOR”, “NUMERO\_PAGINAS” y “PRECIO” (datos relevantes sobre el libro), estos son los principales datos que se tendrían que verificar antes de poder generar el código RDF a partir de las instancias detectadas.

#### *Generación RDF Libro:*

El bloque denominado "Generación RDF" se encarga de transformar automáticamente la información de los libros en un formato RDF, esto sucede cada vez que se identifica una instancia de libro, generando una representación semántica (Figura 5)



```

1,La Chica del Tren,Editorial Planeta,Paula Hawkins,Suspense,Reino Unido,395,13/01/2015,22.99
2,El Código Da Vinci,Editorial Random House,Dan Brown,Suspense,Estados Unidos,489,18/03/2003,20.5

^Z
<rdf:Description rdf:about="#codigoLibro 1">
  <rdf:type rdf:resource="#Libro" />
  <biblioteca:codigoLibro>1</biblioteca:codigoLibro>
  <biblioteca:nombreLibro>La Chica del Tren</biblioteca:nombreLibro>
  <biblioteca:editorial>Editorial Planeta</biblioteca:editorial>
  <biblioteca:autor>Paula Hawkins</biblioteca:autor>
  <biblioteca:genero>Suspense</biblioteca:genero>
  <biblioteca:paisAutor>Reino Unido</biblioteca:paisAutor>
  <biblioteca:numeroPaginas>395</biblioteca:numeroPaginas>
  <biblioteca:anioEdicion>13/01/2015</biblioteca:anioEdicion>
  <biblioteca:precioLibro>22.99</biblioteca:precioLibro>
</rdf:Description>

<rdf:Description rdf:about="#codigoLibro 2">
  <rdf:type rdf:resource="#Libro" />
  <biblioteca:codigoLibro>2</biblioteca:codigoLibro>
  <biblioteca:nombreLibro>El Código Da Vinci</biblioteca:nombreLibro>
  <biblioteca:editorial>Editorial Random House</biblioteca:editorial>
  <biblioteca:autor>Dan Brown</biblioteca:autor>
  <biblioteca:genero>Suspense</biblioteca:genero>
  <biblioteca:paisAutor>Estados Unidos</biblioteca:paisAutor>
  <biblioteca:numeroPaginas>489</biblioteca:numeroPaginas>
  <biblioteca:anioEdicion>18/03/2003</biblioteca:anioEdicion>
  <biblioteca:precioLibro>20.5</biblioteca:precioLibro>
</rdf:Description>

```

Figura 5

Al igual que en el caso de los usuarios tenemos que analizar la instancia que se usa para generar los RDF de los libros, en este caso la estructura se respeta, generando de manera correcta el RDF con la estructura esperada.

**1, La Chica del Tren, Editorial Planeta, Paula Hawkins, Suspense, Reino Unido,395,13/01/2015,22.99**

A partir de la anterior instancia se genera el código RDF correspondiente (Figura 6)

A screenshot of a code editor window with a dark background. The window has three colored window control buttons (red, yellow, green) on the top left and standard window controls (minimize, maximize, close) on the top right. The code is written in a light blue monospaced font and represents an RDF description of a book. The code is as follows:

```
<rdf:Description rdf:about="#codigoLibro 1">
  <rdf:type rdf:resource="#Libro" />
  <biblioteca:nombreLibro>La Chica del Tren</biblioteca:nombreLibro>
  <biblioteca:editorial>Editorial Planeta</biblioteca:editorial>
  <biblioteca:autor>Paula Hawkins</biblioteca:autor>
  <biblioteca:genero>Suspense</biblioteca:genero>
  <biblioteca:paisAutor>Reino Unido</biblioteca:paisAutor>
  <biblioteca:numeroPaginas>395</biblioteca:numeroPaginas>
  <biblioteca:anioEdicion>13/01/2015</biblioteca:anioEdicion>
  <biblioteca:precioLibro>22.99</biblioteca:precioLibro>
</rdf:Description>
```

*Figura 6*

Como podemos ver, la gramática del libro debe de cumplir con un formato especificado para garantizar la conversión exitosa al formato RDF, destacando también que este enfoque de automatización permite procesar múltiples registros de manera automatizada.

### Préstamo

Se definió una estructura para procesar información de los libros para generar datos en formato RDF (Figura 7).



```
// Registros de Préstamos
registrosPrestamos: (prestamo)+;

prestamo:
    numeroPedido=INT COMMA
    codigoLibro=INT COMMA
    codigoUsuario=INT COMMA
    fechaSalida=DATE COMMA
    fechaMaxima=DATE COMMA
    fechaDevolucion=DATE
{
    // Generar RDF para el préstamo
    System.out.println("<!-- Instancia prestamo -->");
    System.out.println("<rdf:Description rdf:about=\"\#numeroPedido "+$codigoLibro.text+"\>");
    System.out.println("    <rdf:type rdf:resource=\"\#Prestamo\" />");
    System.out.println("    <biblioteca:codigoLibro>"+$codigoLibro.text+"</biblioteca:codigoLibro>");
    System.out.println("    <biblioteca:codigoUsuario>"+$codigoUsuario.text+"</biblioteca:codigoUsuario>");
    System.out.println("    <biblioteca:fechaSalida>"+$fechaSalida.text+"</biblioteca:fechaSalida>");
    System.out.println("    <biblioteca:fechaMaxima>"+$fechaMaxima.text+"</biblioteca:fechaMaxima>");
    System.out.println("    <biblioteca:fechaDevolucion>"+$fechaDevolucion.text+"</biblioteca:fechaDevolucion>");
    System.out.println("</rdf:Description>\n");
};
```

Figura 7

En este caso esta gramática es especial ya que de alguna u otra forma llega a relacionar ambas gramáticas anteriores (en cuestión de la estructura), con esto en mente podemos definir por ejemplo los requisitos para poder validar esta gramática, de los cuales son necesarios el “NUMERO DE PEDIDO”, “CODIGO DEL LIBRO”, “CODIGO DEL USUARIO”, FECHA DE SALIDA”, “FECHA MAXIMA” y “FECHA DE DEVOLUCION”, lo que nos muestra cómo se tiene que comportar la estructura de las instancias para la correcta generación del código RDF.

### Generación RDF Libro:

Al terminar las validaciones y el llenado del formato correcto de la instancia podemos empezar la generación del código RDF a partir de las instancias seleccionadas (Figura 8)

```

PS C:\Users\avila\OneDrive\Escritorio\Pruebas> java Test2.java
1,1,101,01/10/2024,15/10/2024,14/10/2024
2,2,102,05/10/2024,20/10/2024,18/10/2024
^Z
<rdf:Description rdf:about="#numeroPedido 1">
  <rdf:type rdf:resource="#Prestamo" />
  <biblioteca:codigoLibro>1</biblioteca:codigoLibro>
  <biblioteca:codigoUsuario>101</biblioteca:codigoUsuario>
  <biblioteca:fechaSalida>01/10/2024</biblioteca:fechaSalida>
  <biblioteca:fechaMaxima>15/10/2024</biblioteca:fechaMaxima>
  <biblioteca:fechaDevolucion>14/10/2024</biblioteca:fechaDevolucion>
</rdf:Description>

<rdf:Description rdf:about="#numeroPedido 2">
  <rdf:type rdf:resource="#Prestamo" />
  <biblioteca:codigoLibro>2</biblioteca:codigoLibro>
  <biblioteca:codigoUsuario>102</biblioteca:codigoUsuario>
  <biblioteca:fechaSalida>05/10/2024</biblioteca:fechaSalida>
  <biblioteca:fechaMaxima>20/10/2024</biblioteca:fechaMaxima>
  <biblioteca:fechaDevolucion>18/10/2024</biblioteca:fechaDevolucion>
</rdf:Description>

```

Figura 8

Aquí podemos ver cómo se genera el código RDF a partir de la instancia del préstamo, incluyendo las propiedades relacionadas con el libro, usuario y las fechas más relevantes del préstamo, como podemos ver en la siguiente instancia:

**1,1,101,01/10/2024,15/10/2024,14/10/2024**

La cual sigue la estructura necesaria para generar correctamente el código RDF del préstamo (Figura 9)

A screenshot of a code editor window with a dark background. The window has three colored window control buttons (red, yellow, green) on the top left and standard window controls (minimize, maximize, close) on the top right. The code is written in a light green monospace font and represents an RDF instance for a library loan.

```
<!-- Instancia prestamo -->
<rdf:Description rdf:about="#numeroPedido 1">
  <rdf:type rdf:resource="#Prestamo" />
  <biblioteca:codigoLibro>1</biblioteca:codigoLibro>
  <biblioteca:codigoUsuario>101</biblioteca:codigoUsuario>
  <biblioteca:fechaSalida>01/10/2024</biblioteca:fechaSalida>
  <biblioteca:fechaMaxima>15/10/2024</biblioteca:fechaMaxima>
  <biblioteca:fechaDevolucion>14/10/2024</biblioteca:fechaDevolucion>
</rdf:Description>
```

*Figura 9*

Este sería solamente un pequeño ejemplo de cómo podríamos facilitar en análisis de los datos de una manera más clara, generando código semántico de manera automática a partir de una representación más simple para el usuario, al igual que la gramática del usuario y del libro, la del préstamo representa más claramente como funcionaria en un ámbito donde ambas están relacionadas.

## Gramática RDF

Como ultima parte de las gramáticas, podemos destacar la ultima de ellas, en este caso esta gramática se encarga de crear un modelo RDF estructurado que puede representar clases y propiedades de un modelo de datos, proporcionando un documento XML listo para ser utilizado en aplicaciones de la web semántica.

```

grammar gramRDF;

inicio: creacion tabla+ cerrar;

// base RDF
creacion:
    CREAM ID {
        System.out.println("<?xml version='1.0'?'>");
        System.out.println();
        System.out.println("<!DOCTYPE rdf:RDF [<!ENTITY xsd 'http://www.w3.org/2001/XMLSchema#'>]>");
        System.out.println();
        System.out.println("<rdf:RDF ");
        System.out.println("    xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'");
        System.out.println("    xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'");
        System.out.println("    xmlns:"+$ID.text+"='http://www.proyecto.com/'+$ID.text+'#'");
        System.out.println("    xml:base='http://www.proyecto.com/'+$ID.text+'#'");
        System.out.println();
    };

// Definición de clases en RDF
tabla:
    TABLA ID INICIO {
        // Inicio de la definición de la clase en RDF.
        System.out.println("<!-- Clase: '"+$ID.text+" -->");
        System.out.println("<rdfs:Class rdf:about='http://www.proyecto.com/biblioteca#'+$ID.text+'/'>");
    } campo+ FIN {
        // Finaliza la tabla (no requiere acción adicional en RDF).
    };

// Definición de propiedades RDF
campo:
    ID (t = NUMERICO | t = ALFABETICO | t = FECHA) {
        System.out.println("<!-- Propiedad: '"+$ID.text+" -->");
        System.out.println("<rdf:Property rdf:about='http://www.proyecto.com/'+$ID.text+'/'>");
        System.out.println("    <rdfs:domain rdf:resource='http://www.proyecto.com/'+$ID.text+'/'>");

        // Asignar el rango correcto según el tipo.
        if (($t.text).compareTo("letras") == 0) {
            System.out.println("    <rdfs:range rdf:resource='http://www.w3.org/2001/XMLSchema#string'>");
        } else if (($t.text).compareTo("numeros") == 0) {
            System.out.println("    <rdfs:range rdf:resource='http://www.w3.org/2001/XMLSchema#integer'>");
        } else if (($t.text).compareTo("fecha") == 0) {
            System.out.println("    <rdfs:range rdf:resource='http://www.w3.org/2001/XMLSchema#date'>");
        }
        System.out.println("</rdf:Property>");
    };

```

```

cerrar:
    CERRAR {
        System.out.println("</rdf:RDF>");
    };

// Tokens
CERRAR: 'cerrar';
NUMERICO: 'numeros';
ALFABETICO: 'letras';
FECHA: 'fecha';
TABLA: 'tabla';
INICIO: 'inicio';
FIN: 'fin';
USAR: 'usar';
CREAR: 'crear';
ID: ('a'..'z' | 'A'..'Z' | '_' ) (
    'a'..'z'
    | 'A'..'Z'
    | '0'..'9'
    | '_'
)*;
WS: (' ' | '\n' | '\t' | '\r')* {$channel=HIDDEN};

```

En primer lugar, como primer punto podemos decir que en la gramática se incluye la declaración XML, namespaces (espacios de nombres) y configuraciones necesarias para que el documento sea válido y entendible en sistemas RDF, se define entidades o conceptos clave del modelo como clases RDF, en donde el objetivo es que se pueda combinar todas las clases y propiedades en un único documento RDF, cerrando la estructura correctamente.

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:biblioteca="http://www.proyecto.com/biblioteca#"
  xml:base="http://www.proyecto.com/biblioteca#"

  <!-- Clase: libro -->
  <rdfs:Class rdf:about="http://www.proyecto.com/biblioteca#libro"/>
  <!-- Propiedad: codigoLibro -->
  <rdf:Property rdf:about="#codigoLibro">
    <rdfs:domain rdf:resource="#codigoLibro"/>
    <rdfs:range rdf:resource="&xsd;integer"/>
  </rdf:Property>
  <!-- Propiedad: nombreLibro -->
  <rdf:Property rdf:about="#nombreLibro">
    <rdfs:domain rdf:resource="#nombreLibro"/>
    <rdfs:range rdf:resource="&xsd:string"/>
  </rdf:Property>
  <!-- Propiedad: editorial -->
  <rdf:Property rdf:about="#editorial">
    <rdfs:domain rdf:resource="#editorial"/>
    <rdfs:range rdf:resource="&xsd:string"/>
  </rdf:Property>
```

Como podemos observar, a partir de la estructura dada podemos generar de manera automática basándonos en el ejemplo de la estructura siguiente:

```
crear biblioteca
tabla libro
  inicio
    codigoLibro numeros
    nombreLibro letras
    editorial letras
    autor letras
    genero letras
    paisAutor letras
    numeroPaginas numeros
    anioEdicion fecha
    precio numeros
  fin
```

Aquí podemos ver como cada instrucción genera fragmentos de código XML que se ensamblan en un archivo completo, generando así el código RDF automático a partir de las instancias dadas.

## RDF Validator

En este caso después del análisis completo de cómo se generaron nuestros códigos RDF, podríamos hacer un análisis en una herramienta como RDF Validator, pero primero tenemos que saber que es, podemos decir que RDF Validator es una herramienta en línea para validar y analizar datos RDF, entonces haremos una pequeña prueba con nuestros datos generados (Figura 10)

Check by Direct Input

```
<rdfs:range rdf:resource="&xsd:string"/>
</rdf:Property>

<rdf:Property rdf:about="#genero">
  <rdfs:domain rdf:resource="#Libro"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</rdf:Property>

<rdf:Property rdf:about="#paisAutor">
  <rdfs:domain rdf:resource="#Libro"/>
  <rdfs:range rdf:resource="#paisAutor"/>
</rdf:Property>
```

Parse RDF Restore the original example Clear the textarea

Display Result Options:  
 Triples and/or Graph: Triples and Graph ▾  
 Graph format: SVG - link ▾

Figura 10

Bajo el contexto anterior podemos decir que la estructura del código es correcta, ya que la herramienta se encarga de validar las estructuras y que tenga sentido, por ejemplo, al hacer la validación nos muestra tres partes (Triples) del código: sujeto, predicado y objeto, los cuales juntos, forman una declaración sobre un recurso (Figura 11).

Number	Subject	Predicate	Object
1	<a href="http://www.proyecto.com/biblioteca#Libro">http://www.proyecto.com/biblioteca#Libro</a>	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>	<a href="http://www.w3.org/2000/01/rdf-schema#Class">http://www.w3.org/2000/01/rdf-schema#Class</a>
2	<a href="http://www.proyecto.com/biblioteca#Prestamo">http://www.proyecto.com/biblioteca#Prestamo</a>	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>	<a href="http://www.w3.org/2000/01/rdf-schema#Class">http://www.w3.org/2000/01/rdf-schema#Class</a>
3	<a href="http://www.proyecto.com/biblioteca#Usuario">http://www.proyecto.com/biblioteca#Usuario</a>	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>	<a href="http://www.w3.org/2000/01/rdf-schema#Class">http://www.w3.org/2000/01/rdf-schema#Class</a>
4	<a href="http://www.proyecto.com/biblioteca#codigoLibro">http://www.proyecto.com/biblioteca#codigoLibro</a>	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property">http://www.w3.org/1999/02/22-rdf-syntax-ns#Property</a>
5	<a href="http://www.proyecto.com/biblioteca#codigoLibro">http://www.proyecto.com/biblioteca#codigoLibro</a>	<a href="http://www.w3.org/2000/01/rdf-schema#domain">http://www.w3.org/2000/01/rdf-schema#domain</a>	<a href="http://www.proyecto.com/biblioteca#Libro">http://www.proyecto.com/biblioteca#Libro</a>
6	<a href="http://www.proyecto.com/biblioteca#codigoLibro">http://www.proyecto.com/biblioteca#codigoLibro</a>	<a href="http://www.w3.org/2000/01/rdf-schema#range">http://www.w3.org/2000/01/rdf-schema#range</a>	<a href="http://www.w3.org/2001/XMLSchema#integer">http://www.w3.org/2001/XMLSchema#integer</a>
7	<a href="http://www.proyecto.com/biblioteca#nombreLibro">http://www.proyecto.com/biblioteca#nombreLibro</a>	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property">http://www.w3.org/1999/02/22-rdf-syntax-ns#Property</a>
8	<a href="http://www.proyecto.com/biblioteca#nombreLibro">http://www.proyecto.com/biblioteca#nombreLibro</a>	<a href="http://www.w3.org/2000/01/rdf-schema#domain">http://www.w3.org/2000/01/rdf-schema#domain</a>	<a href="http://www.proyecto.com/biblioteca#Libro">http://www.proyecto.com/biblioteca#Libro</a>
9	<a href="http://www.proyecto.com/biblioteca#nombreLibro">http://www.proyecto.com/biblioteca#nombreLibro</a>	<a href="http://www.w3.org/2000/01/rdf-schema#range">http://www.w3.org/2000/01/rdf-schema#range</a>	<a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a>
10	<a href="http://www.proyecto.com/biblioteca#editorial">http://www.proyecto.com/biblioteca#editorial</a>	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a>	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property">http://www.w3.org/1999/02/22-rdf-syntax-ns#Property</a>

Figura 11



Después de esta forma podemos ver otra manera de expresar nuestro código RDF, como un “Gráfico de Modelo de Datos”, el cual es una representación visual del modelo RDF, donde los triples se ilustran como nodos y aristas en un grafo dirigido.

El modelo debido a su tamaño puede verse en el siguiente enlace:

[https://www.w3.org/RDF/Validator/ARPServlet.tmp/servlet\\_12699928583722703892.svg](https://www.w3.org/RDF/Validator/ARPServlet.tmp/servlet_12699928583722703892.svg)

En lo anterior podemos ver como los **nodos** son la representación de los sujetos y objetos de los triples, además de también ver las **aristas**, las cuales representan los predicados (propiedades o relaciones) que conectan los nodos.

Con lo anterior definido y probando, podemos llegar a la conclusión de que validar datos RDF, se garantiza que estos puedan ser interpretados y utilizados por diversas plataformas y agentes, mejorando la integración y el intercambio de información en un entorno digital, además de que esta validación fomenta prácticas de desarrollo más robustas y confiables, minimizando errores y mejorando la calidad de los sistemas basados en RDF.

## Integración en IDE

Para mayor practicidad se integraron las gramáticas dentro de un IDE accesible desde un .jar, sea el caso de la gramática para a través de un lenguaje de alto nivel generar el documento en formato RDF se podrá ingresar en un TextArea la entrada de texto o abriendo el archivo (Ilustración 1), para posterior a la compilación, generar el RDF correspondiente (Ilustración 2), la respuesta generada podrá ser copiada al portapapeles directamente a través de un botón.

Misma secuencia para la generación de instancias descritas en el documento, se podrá ingresar el archivo CSV de las instancias (Ilustración 3 y 4) para obtener las instancias generadas en formato RDF (Ilustración 4).



Ilustración 1

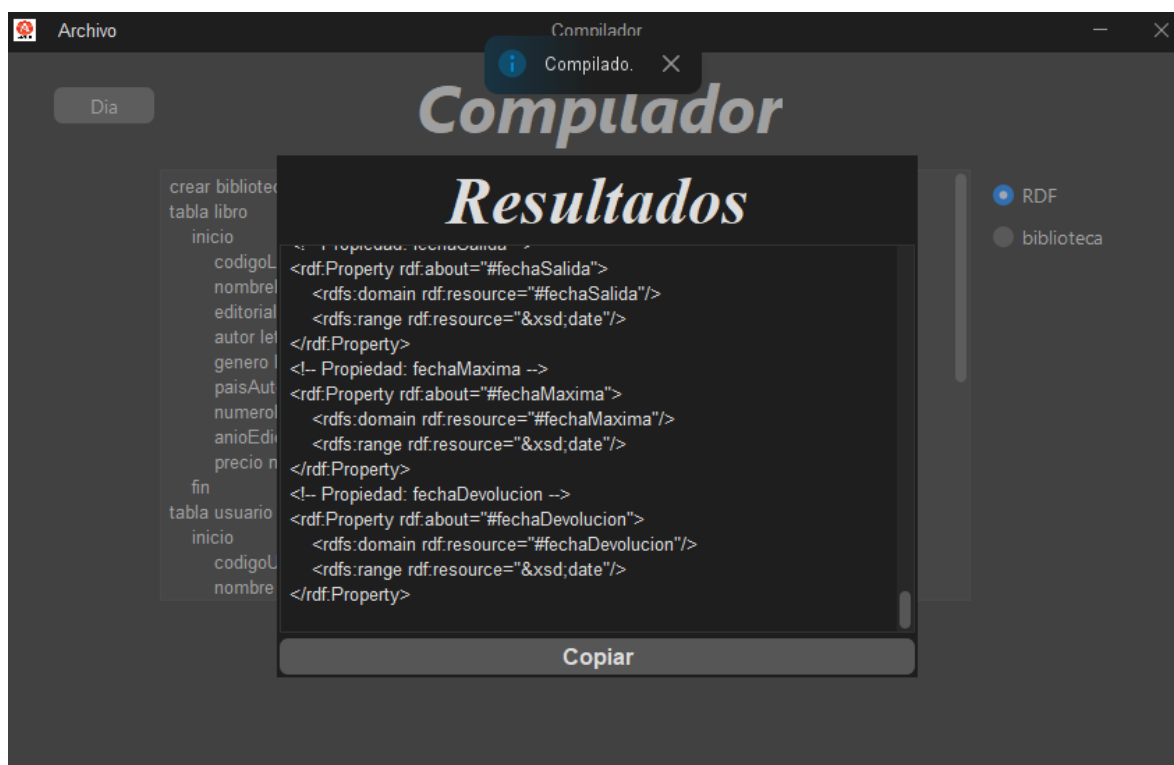


Ilustración 2

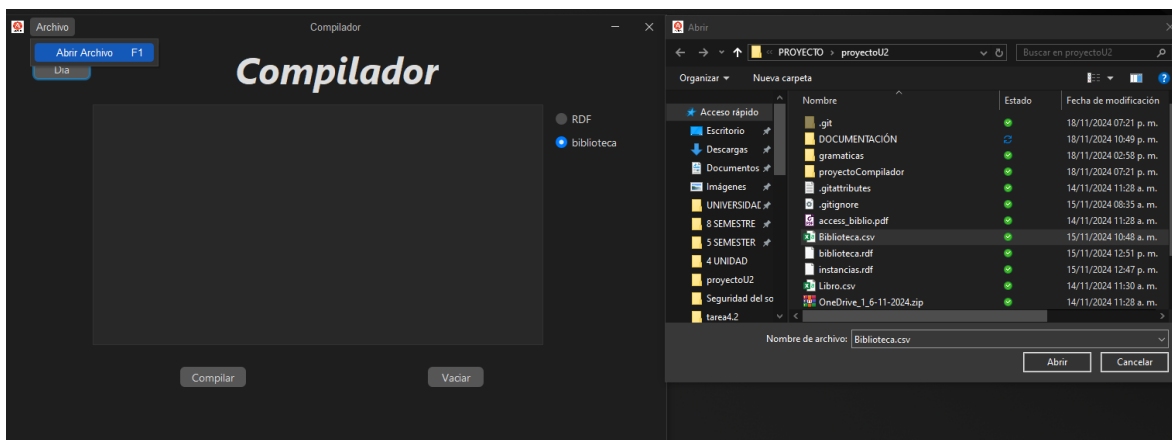


Ilustración 3

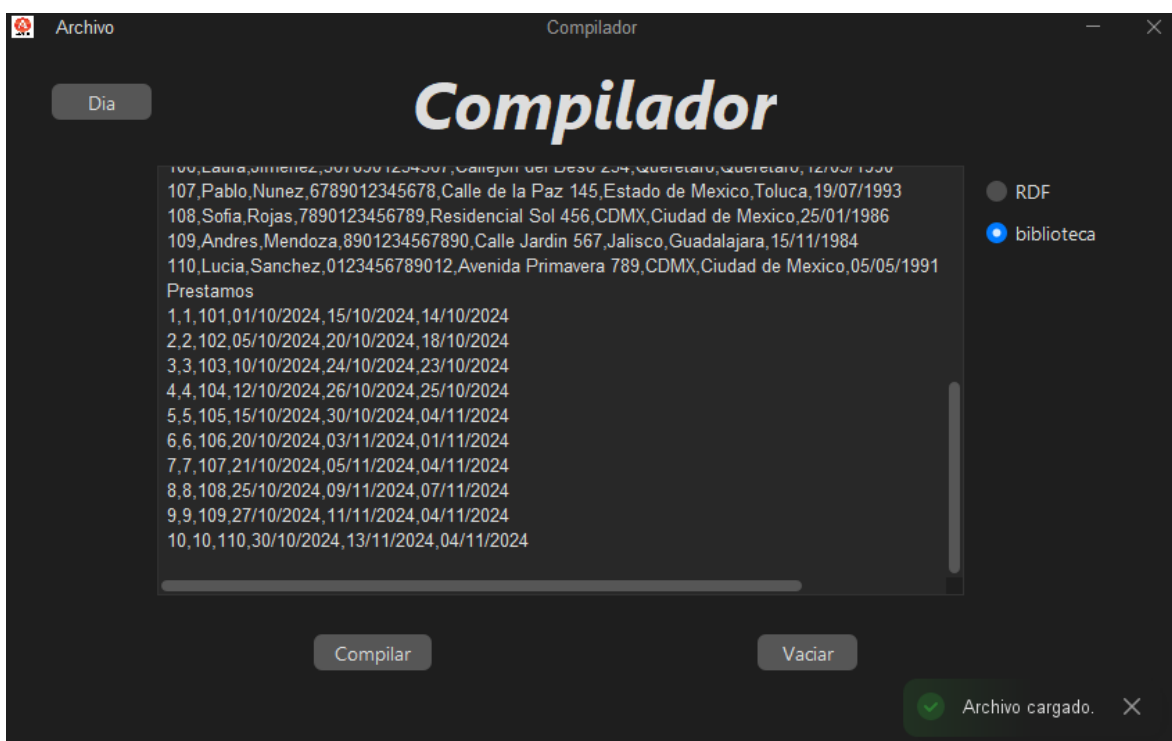


Ilustración 4

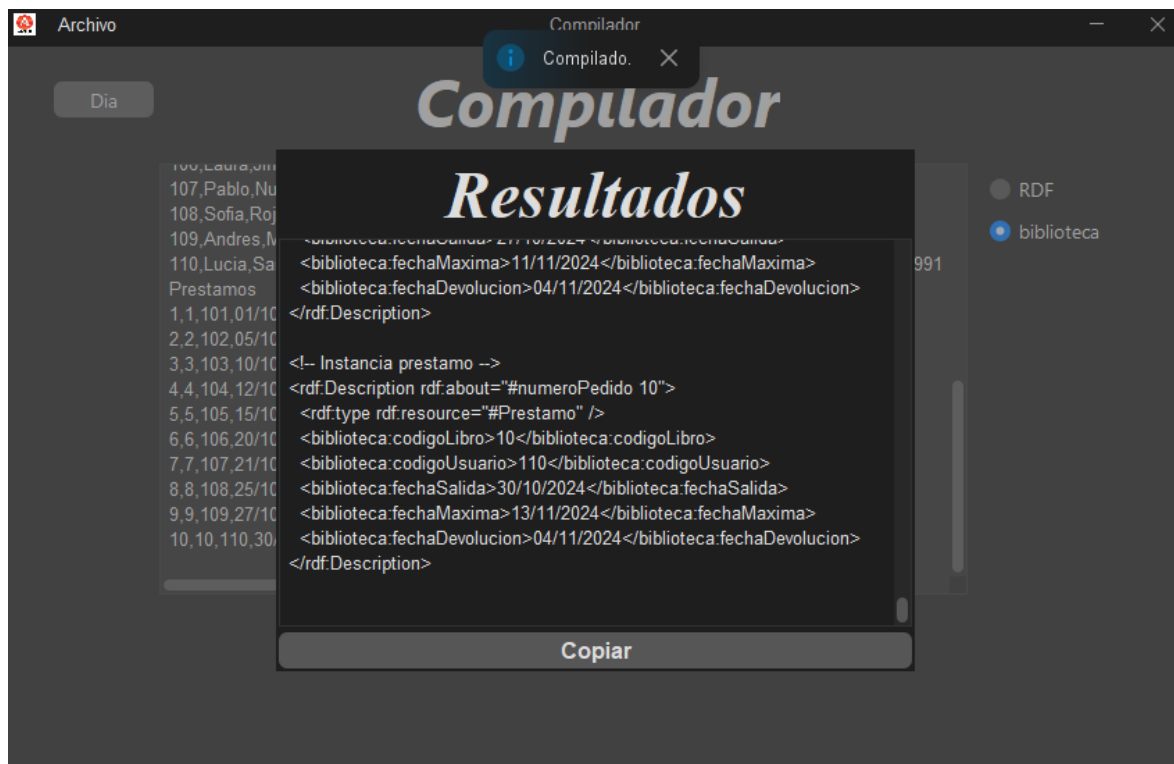


Ilustración 5