Práctica Final
Memoria

Autor:
Adrián Cerros Sánchez - 100405342

Índice

Introducción.....3

Método Revoke Token

Gramáticas.....5

Árboles de derivación.....6

Método Execute Action

Gramáticas.....7

Árboles de derivación.....8

Introducción

El objetivo de la práctica final ha sido desarrollar y testear correctamente las funcionalidades Revoke Token y Execute Action, para ello se ha realizado:

- Desarrollo e implementación del decodificador de Tokens, además de la eliminación del campo Token Value y las correspondientes modificaciones asociadas a este cambio en la implementación anteriormente desarrollada.

- Desarrollo del código y los test para las nuevas funcionalidades del componente. En un primer momento se comenzó su desarrollo guiado por el análisis de clases de equivalencia y valores límite, pero una vez se implementaron los distintos ficheros JSON definidos para los casos de prueba se decidió que el testeo sería mucho más completo si se seguía el desarrollo guiado por pruebas basadas en gramáticas y árboles de derivación, así como la técnica de caminos básicos para otros supuestos dado que las pruebas que se obtenían mediante la técnica de valores límite y clases de equivalencia quedaban totalmente cubiertas si se realizaba testeo de gramática, árboles y estructura añadiendo además muchos más supuestos para las pruebas , por tanto se implementa esta solución.

- Una vez desarrollada la primera funcionalidad (método Revoke Token) se procedió al refactoring de los métodos y clases desarrolladas para favorecer una integración continua en el desarrollo de la siguiente funcionalidad, aplicando el patrón strategy para las clases que efectúan decodificaciones sobre los tokens, lectura de los diferentes ficheros y demás clases que requerían este tipo de solución, desarrollando para cada uno de los casos una interfaz común. Se realizó además una reorganización sobre los paquetes del proyecto para favorecer su comprensión

- Implementación de los casos de prueba y código para la segunda funcionalidad (Execute Action), realizándose de nuevo el mismo procedimiento que se utilizó en la funcionalidad anterior.

- Refactor final sobre las distintas clases del proyecto una vez se tenían desarrolladas todas las funcionalidades y teniendo por tanto una visión global del mismo, se procede a eliminar duplicidades, obtener constantes y reorganizar el código para hacerlo más legible y eficiente, se vuelve a aplicar el patrón strategy donde es necesario.

- Adecuación del proyecto a la normativa de código anteriormente definida, se incluye además una copia de la misma en la carpeta documentation.

- Generación de la documentación de los test y los ficheros jar del proyecto que se encuentran en la carpeta documentation del GIT

- Por último se realiza una revisión sobre la correcta asignación de las dependencias del proyecto maven, así como de la build utilizada por el constructor de maven y la versión del JRE de java, dando el proyecto por finalizado.

Método Revoke Token - Gramática

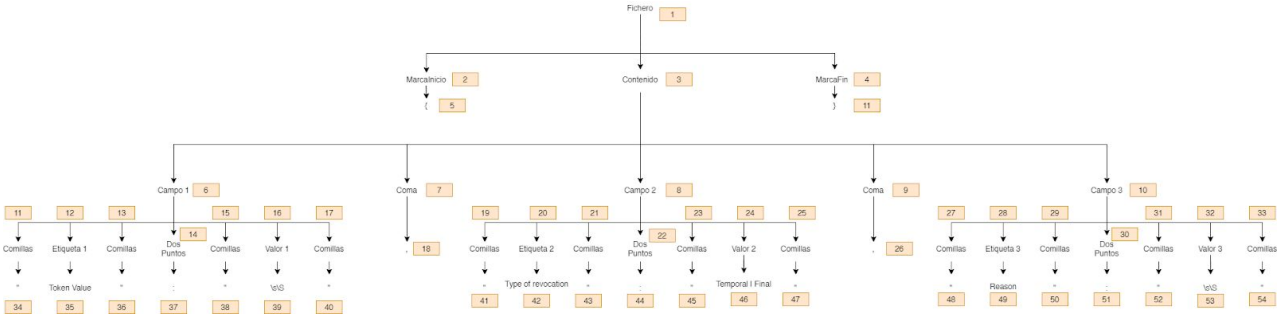
```
{
   "Token Value": "<Cadena de 64 caracteres hexadecimales>",
   "Type of revocation":"<Temporal | Final >",
    "Reason":"<Cadena de 100 caracteres como máximo >"
}
```

La gramática desarrollada para la funcionalidad 4 ha sido la siguiente:

Fichero ::= MarcaInicio Contenido MarcaFin
MarcaInicio ::= '{'
MarcaFin ::= '}'
Contenido ::= Campo1 Coma Campo2 Coma Campo3
Coma ::= ','
Comillas ::= '"'
DosPuntos ::= ':'
Campo1 ::= Comillas Etiqueta1 Comillas DosPuntos Comillas Valor1 Comillas
Etiqueta1 ::= 'Token Value'
Valor1 ::= \s\S
Campo2 ::= Comillas Etiqueta2 Comillas DosPuntos Comillas Valor2 Comillas
Etiqueta2::= 'Type of revocation"
Valor2::= "Temporal" | "Final"
Campo3 ::= Comillas Etiqueta3 Comillas DosPuntos Comillas Valor3 Comillas
Etiqueta3 ::= 'Reason"
Valor3 ::= \s\S {1,100}

Método Revoke Token - Árboles de derivación

El árbol correspondiente a la gramática es el siguiente:



Método Execute Action - Gramática

```
{
   "Token Value": "<Cadena de 64 caracteres hexadecimales>",
   "Type of operation":"<Send Information from Sensor| Send Request to Actuator | Check State >"
}
```

La gramática desarrollada para la funcionalidad 4 ha sido la siguiente:

Fichero ::= MarcaInicio Contenido MarcaFin
MarcaInicio ::= '{'
MarcaFin ::= '}'
Contenido ::= Campo1 Coma Campo2
Coma ::= ','
Comillas ::= '"'
DosPuntos ::= ':'
Campo1 ::= Comillas Etiqueta1 Comillas DosPuntos Comillas Valor1 Comillas
Etiqueta1 ::= 'Token Value'
Valor1 ::=\s\S
Campo2 ::= Comillas Etiqueta2 Comillas DosPuntos Comillas Valor2 Comillas
Etiqueta2::= 'Type of operation"
Valor2::= "Send Information from Sensor" | "Send Request to Actuator" | "Check State"

Método Execute Action - Árboles de derivación

El árbol correspondiente a la gramática es el siguiente: