Seccion3

1

Agenda

- Identificación de Pruebas por Análisis Sintáctico
  - Revisar la especificación de la funcionalidad 2
  - Crear gramática y árbol de derivación
  - Crear Test de pruebas
  - Crear código que cumpla test de pruebas
- Crear cuestiones relativas a la implementación de la funcionalidad

2

RF2

```
{
   "Token Request":"<String having XX characters>",
   "Notification e-mail":"<Valid e-mail address>",
   "Request Date":"<Valid date according to the following format dd/mm/yyyy HH:MM:SS>"
}
```

```
String RequestToken (String InputFile) throws TokenManagementException:
// String represents LM-RF-01-S1
```

3

RF2: Salida un string que comprime la siguiente estructura

Alg	Typ	Device	Request Date	Email	Iat(fecha de emisión)	ex(fecha de expiracion)

HS256	PDS	Device	Request Date	Email	Iat(fecha de emisión)	ex(fecha de expiracion)

En fichero de entrada

Los ponemos nosotros, recomendable usar unix date

Lo encriptamos con SHA-256 (da un hexadecimal de 64)

Lo codificamos en 64urlencoder y este es el String que se devuelve

4

Analisis Sintáctico :Gramática

- 1. Fichero ::= Inicio_objeto Datos Fin_Objeto
- 2. Inicio_objeto ::= {
- 3. Fin:Objeto ::=}
- 4. Datos ::==Campo_TokenRequest Separador Campo_mail Separador Campo_RequestDate
- 5. Campo_TokenRequest::==Etiqueta_TokenRequest Igualdad Valor_TokenRequest
- 6. Campo_mail::==Etiqueta_mail Igualdad Valor_mail
- 7. Campo_RequestDate::==Etiqueta_RequestDate Igualdad Valor_RequestDate
- 8. Etiqueta_ TokenRequest ::= Comillas Valor_ Etiqueta_ TokenRequest Comillas
- 9. Comillas::= “”
- 10. Igualdad::= ‘:’
- 11. Separador ::= ‘,’

5

Analisis Sintáctico :Gramática

- 12. Valor_ Etiqueta_ TokenRequest::= Token Request
- 13. Valor_TokenRequest::= Comillas Valor_TR Comillas
- 14. Valor_TR::= a|b|c|d|e|f|0|1…|9| {32}
- 15. Etiqueta_mail::= Comillas Valor_ Etiqueta_mail Comillas
- 16. Valor_ Etiqueta_mail::= Notification e-mail
- 17. Valor_ mail::= comillas Direccion_eMail comillas
- 18. Direccion_eMail::= nombre_mail arroba dominio punto extensión
- 19. arroba::= “@”
- 20. punto ::=”.”
- 21. nombre_mail::= a…z0..9_-
- 22. dominio = a…z
- 23. extensión = a..z(max 3) …… **se seguiría asi , con el siguiente campo**

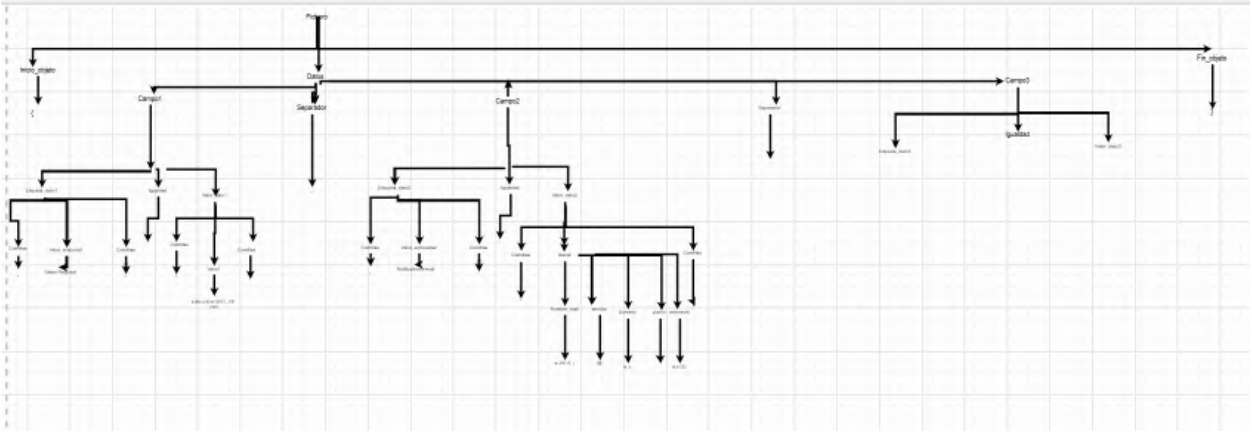
6

Construir árbol de derivación

- Construir el árbol siguiendo la gramática que has hecho
- No te inventes nada , si sigues la gramática que has hecho, detectaras los errores que has cometido en tu gramática.
- Si sigues la gramática , te deberá dar un fichero valido si una vez construido todo el árbol, vas poniendo en un fichero los valores de los nodos no terminales.

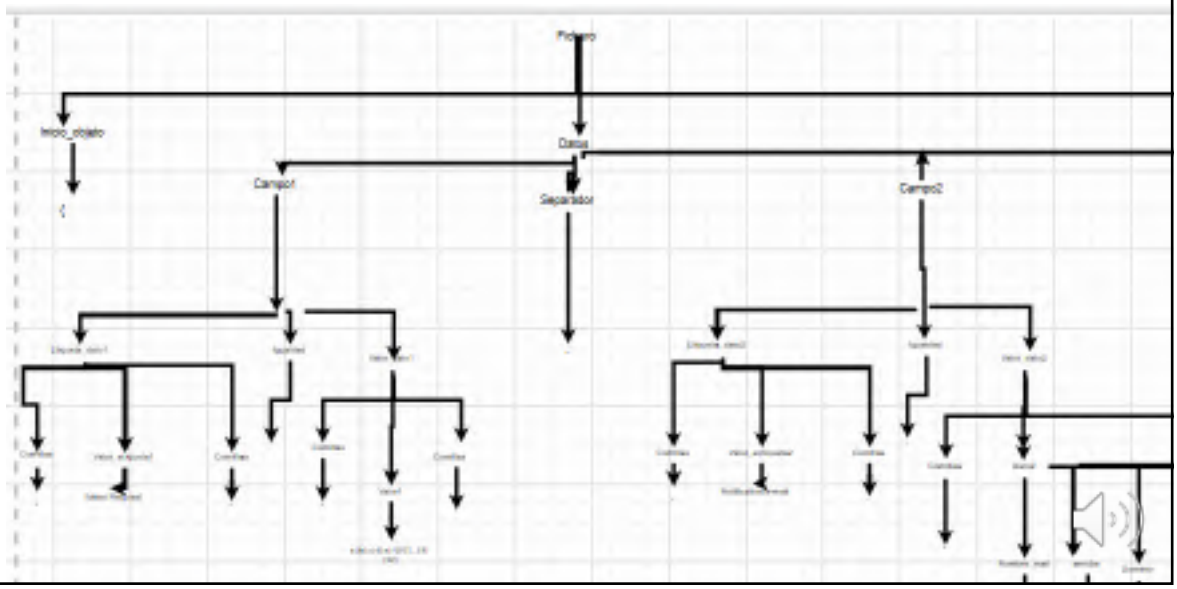
7

Analisis Sintáctico :árbol de derivación (no terminado)



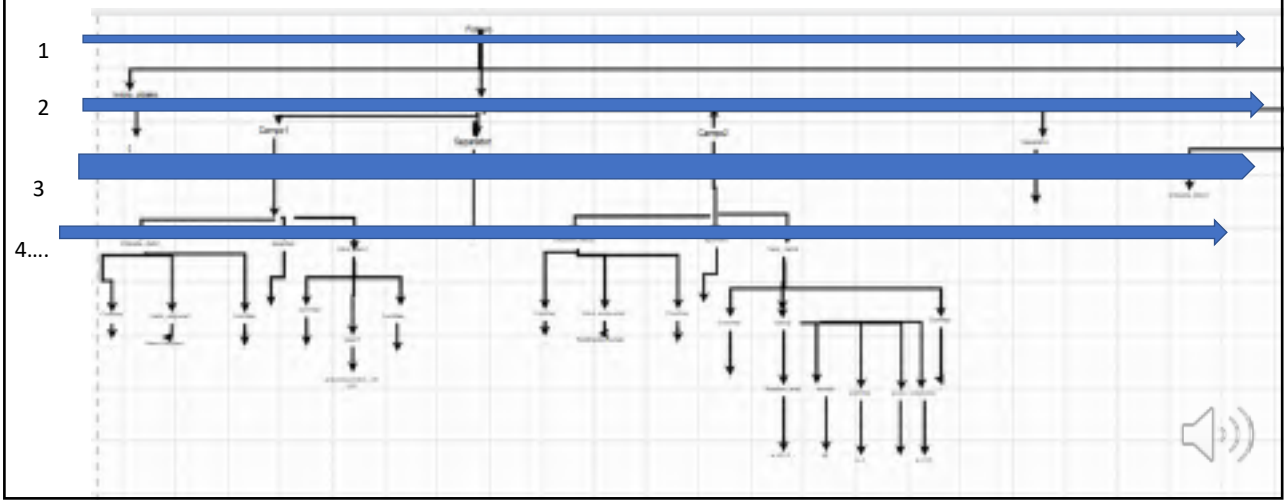
8

Analisis Sintáctico :árbol de derivación : Zoom



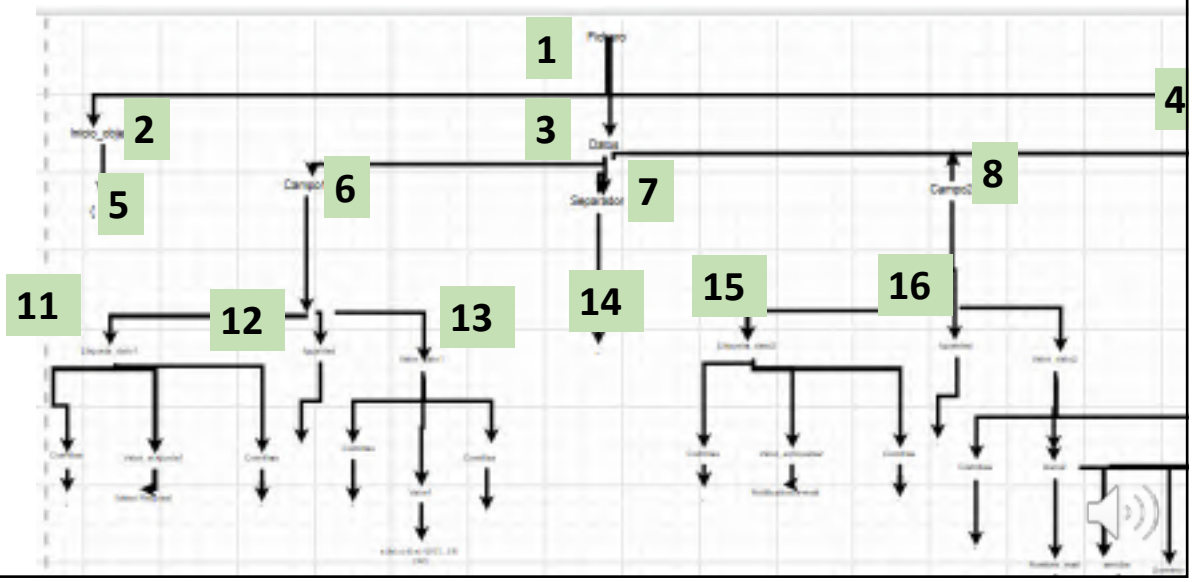
9

Analisis Sintáctico :Numeramos los nodos por niveles (de arriba abajo y de izquierda a derecha)



10

Analisis Sintáctico :árbol de derivación



11

Análisis Sintáctico :Identificación de pruebas – Casos Inválidos

- Casos inválidos:
  - Nodos no terminales: eliminar o añadir nodos
    - Ejemplo: Separadores de campos
      - Eliminar nodo 7 , crear fichero JSON sin ese valor (sin el primer separador)
      - Duplicar nodo 7: crear fichero JSON con ese valor duplicado (con el primer separador duplicado)
  - Nodos terminales : modificar valores a valores invaálidos
    - Modificar nodo 15 : crear fichero JSON poniendo otro valor no valido (por ejemplo, ";")
- Eliminar nodos duplicados en las pruebas
  - Ejemplo : Esta misma situación se debería probar para el separador entre Notification e-Mail y Request Date
    - Sin embargo, al aplicar las reglas de excepción por casos de prueba ya eliminados anteriormente en el árbol de derivación no es necesario considerarlos

12

Análisis Sintáctico :Identificación de pruebas – Casos Inválidos

- Casos inválidos:

```
{
    "Token Request":"f90353fd02ea16b74884d53ae4ebe77"
    "Notification e-mail":"autonomous@vehicle.com",
    "Request Date":"31/07/2019 08:45:59"
}
{
    "Token Request":"f90353fd02ea16b74884d53ae4ebe77”,,
    "Notification e-mail":"autonomous@vehicle.com",
    "Request Date":"31/07/2019 08:45:59"
}
{
    "Token Request":"f90353fd02ea16b74884d53ae4ebe77”;
    "Notification e-mail":"autonomous@vehicle.com",
    "Request Date":"31/07/2019 08:45:59"
}
```

13

Análisis Sintáctico :Identificación de pruebas – Casos Válidos

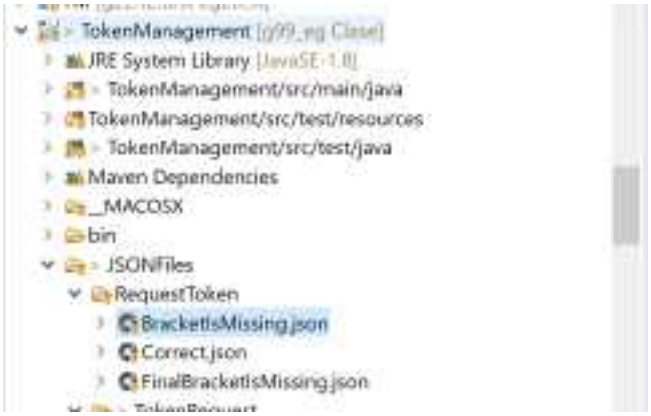
- Casos validos:
  - Primer caso de prueba contemplará todos los nodos NO terminales
  - Se contemplarán casos de prueba adicionales hasta contemplar todos los nodos terminales

```
{
    "Token Request":"f90353fd02ea16b74884d53ae4ebe77",
    "Notification e-mail":"autonomous@vehicle.com",
    "Request Date":"31/07/2019 08:45:59"
}
```

14

Vamos a construir los test de pruebas

Me creo los ficheros JSON para las pruebas y los pongo en una carpeta



15

Creo los Test

- Situándome a nivel de proyecto me creo una clase de test
- Voy a New/Junit Test Case
- En mi caso lo he llamado RequestTokenTest

16

Creamos el test , la parte incial y común a todos los test.

TokenManage... | Token.java | RequestToken... | tokenStore.json | TokenReques... | Correct.json | Brac

```
1  package Transport4Future.TokenManagement;
2
3* import static org.junit.jupiter.api.Assertions.*;
8
9  class RequestTokenTest {
10
11
12     private TokenManager myManager;
13     private String jsonFilesFolder;
14
15*    public RequestTokenTest () {
16         jsonFilesFolder = System.getProperty("user.dir") + "/JSONFiles/RequestToken/";
17
18         myManager = new TokenManager();
19     }
20
```

17

Implementación de Caso de Prueba Inválido

```
@Test
@DisplayName ("Invalid Test Case - JSON File With Two Consecutive Separators")
public void JSONFileWithTwoConsecutiveSeparators () {
        String InputFilePath = System.getProperty("user.dir") +
        "/TestData/TokenRequest/JSONFileWithTwoConsecutiveSeparators.json";
        String expectedMessage = "Error: JSON object cannot be created due to incorrect representation";
        TokenManagementException ex = Assertions.assertThrows(TokenManagementException.class, ()-> {
                myManager.TokenRequestGeneration(InputFilePath);
        });
        assertEquals (expectedMessage,ex.getMessage());
}
```

18

Implementación del Caso de Prueba Válido

```
65
66    @DisplayName("Correct Token Request")
67    @Test
68    void CorrectRequestTokenTest() throws TokenManagementException {
69        String filePath = this.jsonFilesFolder + "Correct.json";
70        String expectedToken = "YmMzMjcyZTIzNGI4MDZhOTBiMDJmMGU0MTkyIjBhYzIxYmQ5MzZiMDVlZWQ2NGJiMmYwYjcyZDQ2ZWMyODY5NQ==";
71        String obtainedToken = myManager.RequestToken(filePath);
72        assertEquals (expectedToken, obtainedToken);
73    }
```

En un test , igual a los del RF1 , pero llamando al nuevo método que hay que realizar.

19

Recomendaciones para el desarrollo de la funcionalidad

- RequestToken debe:
  - Leer fichero json
  - Guardarlo todo en una clase o estructura a la que añadimos algunos campos mas (ver slide mas atrás de como era)
  - Chequear que los valores leídos son validos
  - Crear el string completo
  - Encriptarlo en SHA-256
  - Codificarlo en 64urlencoder
  - Guardarlo en un fichero
  - Devolver el string codificado

20

Creación del método con código inicial para la prueba

```
public String RequestToken(String InputFile) throws TokenManagementException {
    Token myToken = null;
```

21

Creación de la Clase Token

```
package Transport4Future.TokenManagement;

    import java.util.Date;

    public class Token {
        private String alg;
        private String typ;
        private String device;
        private Date requestDate;
        private String notificationEmail;
        private long iat;
        private long exp;

        public Token (String Device, Date RequestDate, String NotificationEmail) {
            this.alg = "HS256";
            this.typ = "PDS";
            this.device = Device;
            this.requestDate = RequestDate;
            this.notificationEmail = NotificationEmail;
//          this.iat = System.currentTimeMillis();
            this.iat = 1583780309;
            this.exp = this.iat + 6048000001;
        }
```

22

Leer fichero json

```
String fileContents = "";


BufferedReader reader;
try {
reader = new BufferedReader(new FileReader(InputFile));
} catch (FileNotFoundException e) {
throw new TokenManagementException("Error: input file not found.");
}
String line;
try {
while ((line = reader.readLine()) != null) {
fileContents += line;
}
} catch (IOException e) {
throw new TokenManagementException("Error: input file could not be accessed.");
}
try {
reader.close();
} catch (IOException e) {
throw new TokenManagementException("Error: input file could not be closed.");
}


// Transform the String with the file contents into a JSON object (in memory).
JsonObject jsonLicense = null;
try {
jsonLicense = Json.createReader(new StringReader(fileContents)).readObject();
}catch (JsonParsingException ex) {
throw new TokenManagementException("Error: JSON object cannot be created due to incorrect representation");
}
```

23

Crear el Objeto Token

```
try {
String tokenRquest = jsonLicense.getString("Token
Request");
String email = jsonLicense.getString("Notification e-
mail");
SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-
dd HH:mm:ss");
Date date = format.parse(jsonLicense.getString("Request
Date"));


myToken = new Token (tokenRquest, date, email);
checkTokenRequestInformationFormat(myToken);
} catch (Exception pe) {
throw new TokenManagementException("Error: invalid input
data in JSON structure.");
}
```

24

Validar el Objeto Token

```
if (TokenToVerify.getDevice().length() != 32) {
throw new TokenManagementException("Error: invalid Device in
token request.");
}
// E-mail RFC822 compliant regex adapted for Java:
Pattern mailPattern = Pattern.compile("(?:(?:\\r\\n)?[
\\t])*(?:(?:(?:[^()<>@,;:\\\\\".\\[\\] \\000-\\031”);
if
(!mailPattern.matcher(TokenToVerify.getNotificationEmail()).match
es()) {
throw new TokenManagementException("Error: invalid E-mail data in
JSON structure.");
}
```

25

Generar Hash

```
public String CodeHash256(Token myToken) throws TokenManagementException {
MessageDigest md;
try {
    md = MessageDigest.getInstance("SHA-256");
} catch (NoSuchAlgorithmException e) {
    throw new TokenManagementException("Error: no such hashing algorithm.");
}

String input = "Stardust" + "-" + myToken.toString();

md.update(input.getBytes(StandardCharsets.UTF_8));
byte[] digest = md.digest();

// Beware the hex length. If MD5 -> 32:"%032x", but for instance, in SHA-256 it should be "%064x"
String hex = String.format("%064x", new BigInteger(1, digest));

return hex;
}
```

26

Codificar String en 64 url encoder

```
private String encodeString(String stringToEncode) throws TokenManagementException {

String encodedURL;

try {
    encodedURL = Base64.getUrlEncoder().encodeToString(stringToEncode.getBytes());
}catch (Exception ex)
{
    throw new TokenManagementException("Error encoding 64URL.");
}

    return encodedURL;
}
```

27

Guardar en un fichero

```
TokensStore myStore = new TokensStore ();
myStore.Add(myToken);
```

28

Guardar en fichero – Clase Token Store

```
1  package Transport4Future.TokenManagement;
2
3  import java.io.FileReader;
12
13 public class TokensStore {
14
15     private List<Token> tokensList;
16
17     private void Load () {
18         try
19         {
20             JsonReader reader = new JsonReader(new FileReader(System.getProperty("user.dir") + "/Store/tokenStore.json"));
21             Gson gson = new Gson();
22             Token [] myArray = gson.fromJson(reader, Token[].class);
23             this.tokensList = new ArrayList<Token>();
24             for (Token token: myArray) {
25                 this.tokensList.add(token);
26             }
27         }
28         catch (Exception ex)
29         {
30             this.tokensList = new ArrayList<Token>();
31         }
32     }
33
```

29

Guardar fichero fichero – Clase Token Store

```
39
40     public void Add (Token newToken) throws TokenManagementException {
41         this.Load();
42         if (Find(newToken.toString())==null) {
43             tokensList.add(newToken);
44             this.Save();
45         }
46     }
47
48     private void Save () throws TokenManagementException {
49         Gson gson = new Gson();
50         String jsonString = gson.toJson(this.tokensList);
51         FileWriter fileWriter;
52         try {
53             fileWriter = new FileWriter(System.getProperty("user.dir") + "/Store/tokenStore.json");
54             fileWriter.write(jsonString);
55             fileWriter.close();
56         } catch (IOException e) {
57             throw new TokenManagementException("Error: Unable to save a new token in the internal licenses store");
58         }
59     }
60
```

30

Guardar fichero – Clase Token Store

```
    public Token Find (String tokenToFind) {
        Token result = null;
        this.Load();
        for (Token token : this.tokensList) {
            if (token.toString().equals(tokenToFind)) {
                result = token;
            }
        }
        return result;
    }
}
```

31

Herramientas

- Draw io
  - https://www.draw.io/
  - Herramienta sencilla de manejar.
  - Lo salva como XML para que luego lo puedas exportar o volver a editar . Tambien se puede hacer un print a PDF .

32