Ejercicio Guiado 3 – Sesión 3 – Capítulo 5

Agenda de hoy

- Repasar el funcionamiento esperado de la Funcionalidad 3 del EG3
- Presentar el código desarrollado para su implementación
- Prueba mediante la prueba de los caminos básicos
  - Dibujar el flujo de control
  - Identificar los caminos básicos
  - Identificar y Crear casos de prueba
- Prueba mediante el análisis de las estructuras de control
  - Análisis de Estructuras repetitivas
  - Identificar y Crear Casos de Prueba

Token Management – Requisito Funcional 3

```
boolean VerifyToken (String Token) throws TokenManagementException;
// boolean represents LM-RF-03-S1
// String Token represents LM-RF-03-E1
// TokenManagerException represents LM -RF-03-S2
```

Creo el método VerifyToken

- VerifyToken debe:
  1. Acceder al repositorio local de Tokens emitidos
  2. Comprobar que el Token es uno de los que tenemos registrado
  3. Comprobar que el Token está activo y su fecha de expiración no ha pasado.
  4. Devolver true o false según corresponda

Acceder al repositorio local de Tokens emitidos

```
public boolean VerifyToken (String Token) throws TokenManagementException{
    boolean result = false;
    TokensStore myStore = new TokensStore ();
```

Comprobar que el Token es uno de los que tenemos registrado -1

```
public boolean VerifyToken (String Token) throws TokenManagementException{
    boolean result = false;
    TokensStore myStore = new TokensStore ();

    Token tokenFound = myStore.Find(Token);
```

Comprobar que el Token es uno de los que tenemos registrado -2

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

Comprobar que el Token es uno de los que tenemos registrado -3

```
private void Load () {
    try
    {
        JsonReader reader = new JsonReader(new FileReader(System.getProperty("use
        Gson gson = new Gson();
        Token [] myArray = gson.fromJson(reader, Token[].class);
        this.tokensList = new ArrayList<Token>();
        for (Token token: myArray) {
            this.tokensList.add(token);
        }
    }
    catch (Exception ex)
    {
        this.tokensList = new ArrayList<Token>();
    }
}
```

Comprobar que el Token es uno de los que tenemos registrado -4

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

Comprobar que el Token es uno de los que tenemos registrado -5

```
public class Token {
    private String alg;
    private String typ;
    private String device;
    private String requestDate;
    private String notificationEmail;
    private long iat;
    private long exp;
    private String signature;
    private String tokenValue;
```

```
public void setTokenValue(String value) {
    this.tokenValue = value;
}

public String getTokenValue() {
    return this.tokenValue;
}
```

Comprobar que el Token es uno de los que tenemos registrado -6

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

Comprobar que el Token está activo y su fecha de expiración no ha pasado -1

```
public boolean VerifyToken (String Token) throws TokenManagementException{
    boolean result = false;
    TokensStore myStore = new TokensStore ();

    Token tokenFound = myStore.Find(Token);

    if (tokenFound!=null){
        result = isValid(tokenFound);
    }
    return result;
}
```

Comprobar que el Token está activo y su fecha de expiración no ha pasado -2

```
public class TokenManager

    private boolean isValid (Token tokenFound) {
        if ((!tokenFound.isExpired()) && (tokenFound.isGranted())){
            return true;
        }
        else {
            return false;
        }
    }
```

Comprobar que el Token está activo y su fecha de expiración no ha pasado -3

```
public class Token {

    public boolean isExpired () {
        if (this.exp > System.currentTimeMillis()) {
            return false;
        }
        else {
            return true;
        }
    }
```

Comprobar que el Token está activo y su fecha de expiración no ha pasado -4

```
public class TokenManager

    private boolean isValid (Token tokenFound) {
        if ((!tokenFound.isExpired()) && (tokenFound.isGranted())){
            return true;
        }
        else {
            return false;
        }
    }
```

Comprobar que el Token está activo y su fecha de expiración no ha pasado -5

```
public class Token {

    public boolean isGranted () {
        if (this.iat < System.currentTimeMillis()) {
            return true;
        }
        else {
            return false;
        }
    }
```

Devolver true o false según corresponda

```
public boolean VerifyToken (String Token) throws TokenManagementException{
    boolean result = false;
    TokensStore myStore = new TokensStore ();

    Token tokenFound = myStore.Find(Token);

    if (tokenFound!=null){
        result = isValid(tokenFound);
    }
    return result;
}
```

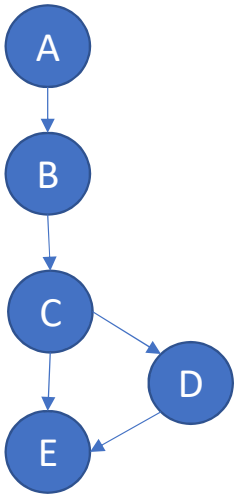
Técnica de Prueba de Camino Básico

1. Derivar el grafo del flujo de control de un método a probar
2. Calcular la complejidad ciclomática del grafo
3. Seleccionar el valor obtenido como el número de caminos distintos a probar
4. Crear un caso de prueba para la ejecución de cada camino
5. Ejecutar estos casos de prueba

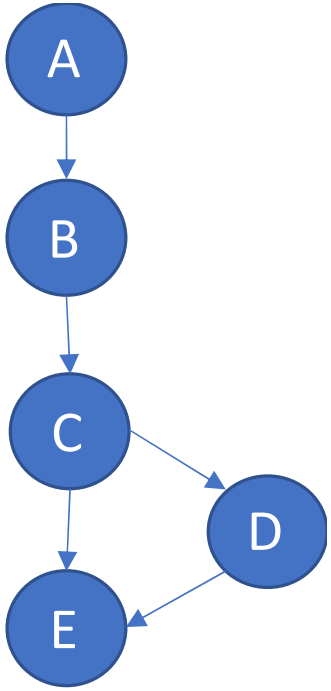
Técnica de Análisis estructural – Dibujar el grafo de Control

```
public boolean VerifyToken (String Token) throws TokenManagementException{
    boolean result = false;
    TokensStore myStore = new TokensStore ();

    Token tokenFound = myStore.Find(Token);

    if (tokenFound!=null){
        result = isValid(tokenFound);
    }
    return result;
}
```

A
B
C
D
E



Técnica de Análisis estructural – Complejidad e identificación de caminos básicos



V(G) = Enlaces – Nodos + 2

V(G) = 5 – 5 + 2 = 2

Camino 1: A, B, C, E

Pasar como parámetro un token que no existe

Camino 2: A, B, C, D, E

Pasar como parámetro un token no existe ya sea válido o no (habrá que profundizar en estos casos)

Técnica de Análisis estructural – Crear un caso de prueba para la ejecución de cada camino

```
@Test
@DisplayName("RF03 - TC01 - Buscar un token que no existe")
void VerifyTokenNonExistingToken() throws TokenManagementException {
    String tokenToVerify = "ABxnPUhTMjU2XG4gVHlwPVBEU1xuRGV2PTI3OGU3ZTI3NzMy
    boolean result = myManager.VerifyToken(tokenToVerify);
    assertEquals (result,false);
}
```

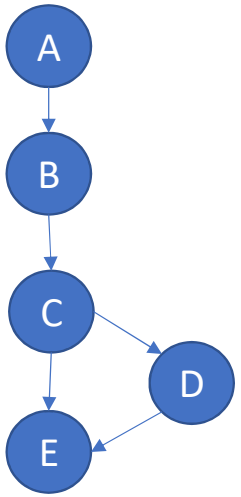
Técnica de Análisis estructural – Crear un caso de prueba para la ejecución de cada camino

```
@Test
@DisplayName("RF03 - TC02 - Buscar un token que existe en nuestros registros")
void VerifyTokenCorrectTest() throws TokenManagementException {
    String tokenToVerify = "QWxnPUhTMjU2XG4gVHlwPVBEU1xuRGV2PTI3OGU3ZTI3NzMyYzRlN
    boolean result = myManager.VerifyToken(tokenToVerify);
    assertEquals (result,false);
}
```

Técnica de Análisis estructural – Dibujar el grafo de Control

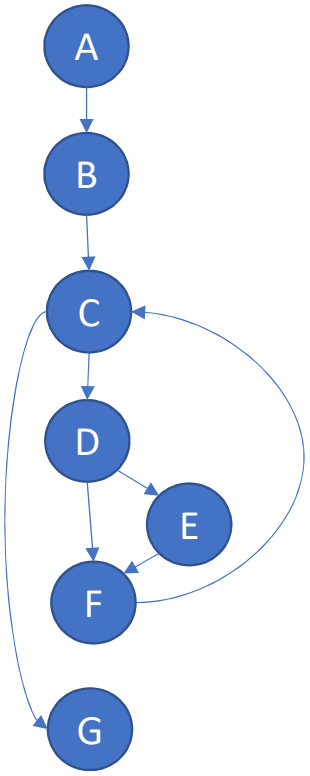
```
public boolean VerifyToken (String Token) throws TokenManagementException{
    boolean result = false;
    TokensStore myStore = new TokensStore ();

    Token tokenFound = myStore.Find(Token); (*)

    if (tokenFound!=null){
        result = isValid(tokenFound);
    }
    return result;
}
```

A
B
C
D
E

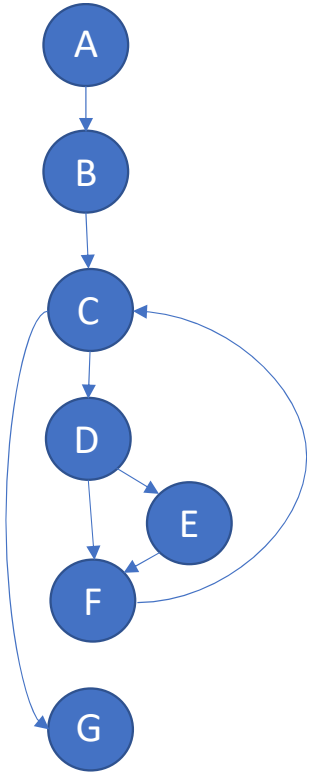


Técnica de Análisis estructural – Dibujar el grafo de Control

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



Técnica de Análisis estructural – Complejidad e identificación de caminos básicos



V(G) = Enlaces – Nodos + 2

V(G) = 8 – 7 + 2 = 3

Camino 1: A, B, C, G

Búsqueda de un token con el almacén vacío

Camino 2: A, B, C, D, F, G

El almacén de tokens tiene un registro, pero busco un registro que no existe

Camino 3: A, B, C, D, E, C, G

El almacén de tokens tiene un registro, pero busco un registro que no existe

Técnica de Análisis estructural – Crear un caso de prueba para la ejecución de cada camino

```
private void resetTokenStore () throws TokenManagementException {
    String storePath = System.getProperty("user.dir") + "/Store/tokenStore.json";
    FileWriter fileWriter;
    try {
        fileWriter = new FileWriter(storePath);
        fileWriter.close();
    } catch (IOException e) {
        throw new TokenManagementException("Error: Unable to save a new token in
    }
}
```

```
@Test
@DisplayName("RF03 - TC01 - Buscar un token en un almacén de tokens vacío")
void VerifyTokenEmptyTokenStore() throws TokenManagementException {
    this.resetTokenStore();
    String tokenToVerify = "ABxnPUhTMjU2XG4gVHlwPVBEU1xuRGV2PTI3OGU3ZTI3NzMyYzRl
    boolean result = myManager.VerifyToken(tokenToVerify);
    assertEquals (result,false);
}
```

Técnica de Análisis estructural – Crear un caso de prueba para la ejecución de cada camino

```
private void insertFirstToken () throws TokenManagementException {
    this.resetTokenStore();
    String InputFile = System.getProperty("user.dir") + "/TestData/TokenRequestT
    myManager.RequestToken(InputFile);
}

@Test
@DisplayName("RF03 - TC02 - Buscar un token que no existe en un almacén de regi
void VerifyTokenNonExistingToken() throws TokenManagementException {
    this.insertFirstToken();
    String tokenToVerify = "ABxnPUhTMjU2XG4gVHlwPVBEU1xuRGV2PTI3OGU3ZTI3NzMyYzR
    boolean result = myManager.VerifyToken(tokenToVerify);
    assertEquals (result,false);
}
```

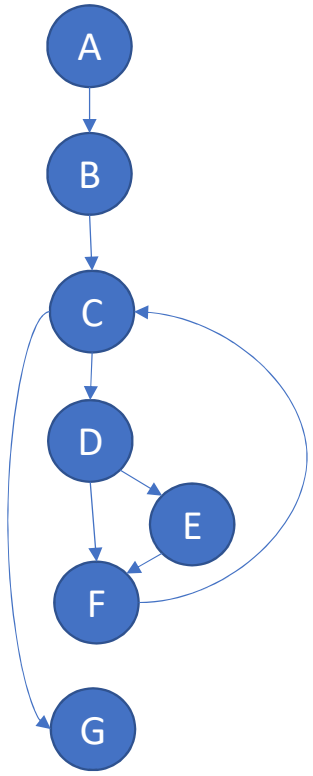
Técnica de Análisis estructural – Crear un caso de prueba para la ejecución de cada camino

```
@Test
@DisplayName("RF03 - TC03 - Buscar un token que existe en almacén con registros")
void VerifyTokenCorrectTest() throws TokenManagementException {
    this.insertFirstToken();
    String tokenToVerify = "QWxnPUhTMjU2XG4gVHlwPVBEU1xuRGV2PTI3OGU3ZTI3NzMyYzRlN
    boolean result = myManager.VerifyToken(tokenToVerify);
    assertEquals (false,result);
}
```

Técnica de Análisis estructural – Dibujar el grafo de Control

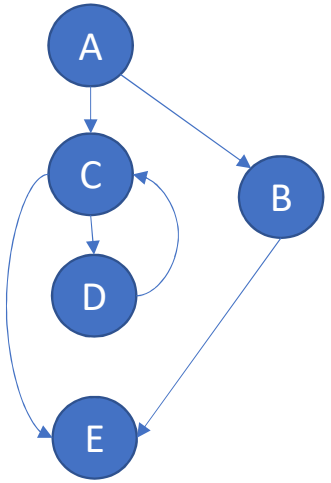
```
public Token Find (String tokenToFind) {
    Token result = null;
    this.Load(); (*)
    for (Token token : this.tokensList) {
        if (token.getTokenValue().equals(tokenToFind)) {
            result = token;
        }
    }
    return result;
}
```

A
B
C
D
E
F
G

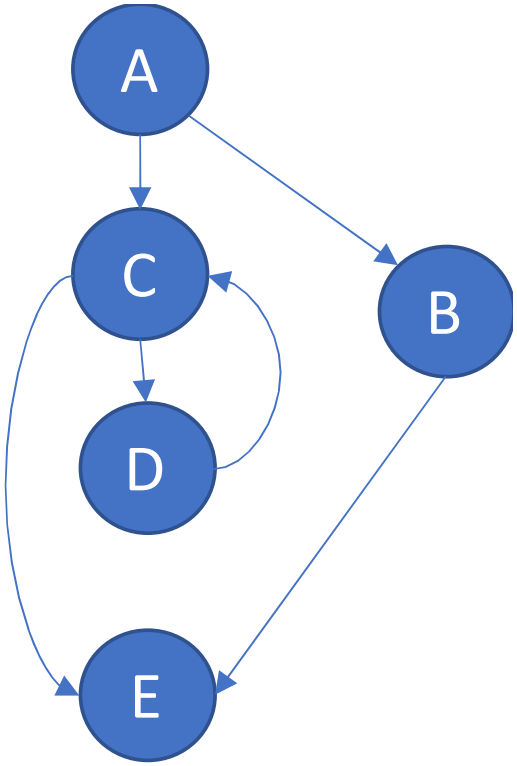


Técnica de Análisis estructural – Dibujar el grafo de Control

```
private void Load () {
    try
    {
A       JsonReader reader = new JsonReader(new FileReader(System.getProperty("use
        Gson gson = new Gson();
        Token [] myArray = gson.fromJson(reader, Token[].class);
        this.tokensList = new ArrayList<Token>();
C       for (Token token: myArray) {
            this.tokensList.add(token);
D
E       }
    }
    catch (Exception ex)
    {
        this.tokensList = new ArrayList<Token>();
B
    }
}
F
```



Técnica de Análisis estructural – Complejidad e identificación de caminos básicos



V(G) = Enlaces – Nodos + 2

V(G) = 6 – 5 + 2 = 3

Camino 1: A, B, E

Fichero Token Store No Existe

Camino 2: A, C, E

El almacén de tokens existe pero está vacío

Camino 3: A, C, D, C, E

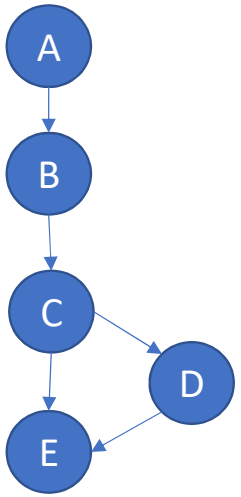
El almacén de tokens tiene, al menos, un registro

Técnica de Análisis estructural – Crear un caso de prueba para la ejecución de cada camino

```
@Test
@DisplayName("RF03 - TC04 - Buscar un token cuando el almacén no existe")
void VerifyTokenNonExitingTokenStore() throws TokenManagementException {
    String storePath = System.getProperty("user.dir") + "/Store/tokenRequestsStor
    File file = new File(storePath);
    file.delete();
    String tokenToVerify = "ABxnPUhTMjU2XG4gVHlwPVBEU1xuRGV2PTI3OGU3ZTI3NzMyYzRlN
    boolean result = myManager.VerifyToken(tokenToVerify);
    assertEquals (result,false);
}
```

Técnica de Análisis estructural – Dibujar el grafo de Control

```
public boolean VerifyToken (String Token) throws TokenManagementException{
    boolean result = false;                                          A
    TokensStore myStore = new TokensStore ();

    Token tokenFound = myStore.Find(Token);                          B

    if (tokenFound!=null){                                           C
        result = isValid(tokenFound);                                D  (*)
    }
    return result;                                                   E
}
```



Técnica de Análisis estructural – Dibujar el grafo de Control

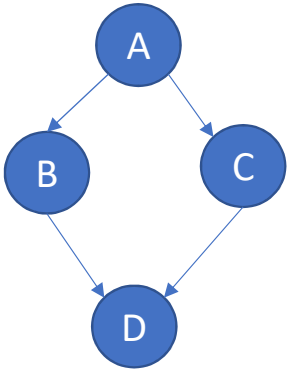
```
private boolean isValid (Token tokenFound) {
    if ((!tokenFound.isExpired()) && (tokenFound.isGranted())){
        return true;
    }
    else {
        return false;
    }
}
```

A

B

C

D



V(G) = Enlaces – Nodos + 2
V(G) = 4 – 4 + 2 = 2

Camino 1: A, B, D Token Válido

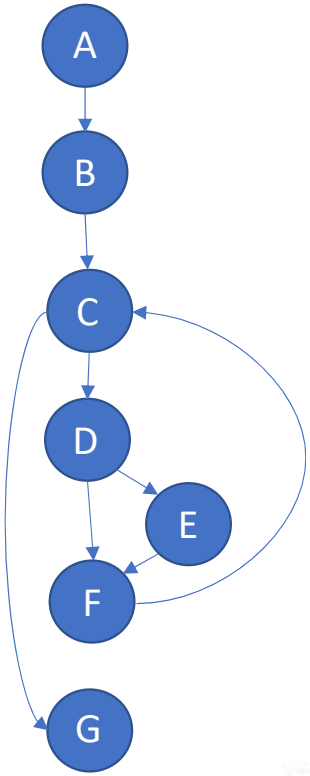
Camino 2: A, C, E Token Expirado

Técnica de Análisis estructural – Crear un caso de prueba para la ejecución de cada camino

```
@Test
@DisplayName("RF03 - TC05 - Buscar un token válido")
void VerifyValidToken() throws TokenManagementException {
    this.insertFirstToken();
    this.insertSecondToken();
    String tokenToVerify = "QWxnPUhTMjU2XG4gVHlwPVBEU1xuRGV2PTI3OGU3ZTI3NzMyYzI
    boolean result = myManager.VerifyToken(tokenToVerify);
    assertEquals (result,true);
}
```

Prueba exhaustiva de estructuras repetitivas

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



Identificación de casos de prueba

- No pasar por el bucle – Caso de prueba sin ningún registro en Token Store – TC01
- Pasar por el bucle 1 vez – Caso de prueba con 1 registro en Token Store – TC02
- Pasar por el bucles 2 veces - Caso de prueba con 2 registros en Token Store– TC05
- Pasar por el bucle max-1 veces
- Pasar por el bucle el número máx. de veces
- Intentar pasar por el bucle máx.+1 veces

} Probar con un elevado número de casos, lo que permita el almacenamiento

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 .