Universidad Carlos III

Curso Desarrollo de Software 2021-22

Curso 2021-22

Práctica Final

Fecha: **19/05/22**

GRUPO: **80** Equipo: **06** NIA: **100451189 / 100451139**

Alumnos: **Jorge Álvarez Rodríguez / Alba Marco Ugarte**

Correos: **100451189@alumnos.uc3m.es / 100451139@alumnos.uc3m.es**

TABLA DE CONTENIDO

1. Introducción

La práctica proporciona un código inicial con el que se plantea un sistema de gestión de vacunas y citas para cada una de las comunidades autónomas del país. De acuerdo a esto, se pide llevar a cabo una serie de modificaciones para poder implementar mejoras en dicho sistema y hacer que los procesos de control sean más efectivos.
En esta memoria se expondrán explicaciones de cada uno de los cambios realizados sobre los métodos indicados y sobre el método nuevo pedido para el correcto desarrollo de la práctica. Junto a dichas justificaciones de la funcionalidad, se incluirá información que permita el correcto entendimiento del proceso TDD llevado a cabo.

2. Descripción del código implementado

A continuación, se explica el código que ha sido implementado y algunos de los elementos requeridos por el propio enunciado de la práctica.

2.1 Método get_vaccine_date

2.1.1 Funcionalidad

En la clase VaccineManager hay un método implementado que recibe el nombre de "get_vaccine_date". Dicho método tiene como objetivo crear una cita para la fecha indicada por parámetro. Este método recibe como argumentos un fichero en formato JSON y una fecha (posee un formato ISO: "yyyy-mm-dd") en la que le gustaría al paciente establecer una cita. Para la correcta ejecución del método, se han implementado una serie de comprobaciones para verificar que la cita se puede generar exitosamente. Entre estas comprobaciones se encuentran comprobar que la fecha introducida es válida, es decir, que no sea anterior ni igual a la fecha actual. En caso de que nada pueda generar un posible error, la cita se creará de manera correcta.

2.1.2 Casos de prueba aplicados

Los tests que se han llevado a cabo para verificar el correcto funcionamiento del método implementado son los siguientes:

1) test_get_vaccine_date_menor: se encarga de comprobar que recibiendo por parámetro una fecha anterior a la fecha actual, se detecta un problema y la cita no es aceptada para su creación.
2) test_get_vaccine_date_mayor: encargado de verificar que recibiendo como parámetro una fecha posterior a la fecha actual, no ocurre ningún error y la cita es permitida y creada correctamente.
3) test_get_vaccine_date_igual: se encarga de comprobar que recibiendo por parámetro la misma fecha a la actual, salta un error (VaccineManagementException) y no se permite la creación de la cita deseada.
4) test_get_vaccine_date_bad_format: se encarga de verificar que al recibir una fecha con un formato incorrecto al establecido ("yyyy-mm-dd"), el proceso se interrumpe y la creación de la cita no es llevada a cabo.

Los primeros tres tests corresponden a aquellos empleados para la técnica de **valores límite**, ya que se contempla un caso por debajo del límite (fecha anterior), un caso en el límite (fecha actual) y una fecha por encima del límite (fecha posterior). En cuanto, al último test, relacionado con el formato de la fecha, se asigna como un test que hace uso de la técnica de **clases de equivalencia**.

2.2 Método cancel_appointment

2.2.1 Funcionalidad

En la clase VaccineManager se ha implementado un nuevo método llamado "cancel_appointment", cuyo objetivo es anular una cita predefinida. Éste recibe como parámetro un fichero en formato JSON. Dicho fichero tiene como contenido una serie de valores. Entre ellos encontramos la"date_signature", que consiste en un valor hexadecimal de 64 caracteres; "cancelation_type", que puede tomar los valores "Final" (una cita con este valor se mantiene registrada para posibles futuras funcionalidades del código) y "Temporal" (una cita que toma este valor indica que podría ser reactivada de nuevo); y, por último, "reason", que consiste en una descripción (*string*) con una longitud mínima de dos caracteres y máxima de cien.

En primer lugar, esta función contempla si el contenido pasado como parámetros de entrada es válido. Una vez verificado que todo está correcto, comprueba posibles casos de error. Estos casos de errores pueden ser los siguientes: la cita indicada en el fichero JSON recibido no se encuentra registrada y por tanto, no existe; la fecha de la cita recibida es previa a la actual, lo cual indica que ya ha pasado; el paciente correspondiente a la cita ya ha sido vacunado; o la cita ya había sido cancelada anteriormente. Por último, al haber llevado a cabo todo el proceso de verificación y creación de una cancelación, el método devuelve el "date_signature" de la cita que acaba de ser anulada.

2.2.2 Casos de prueba aplicados

Para la creación de los casos de prueba en base al diagrama de flujo planteado posteriormente, se han realizado todos los posibles caminos salvo los siguientes: caminos 1,2; 1,3,4; 1,3,5,6; 1,3,5,7,8; entre otros. Dichos caminos ya se ven contemplados en tests implementados anteriormente en el código realizado para la gramática. Los casos de prueba que se han implementado son los siguientes:

1) test4_1_not_ok: Recorre los nodos en el siguiente orden: 1, 3, 5, 7, 9, 10, 11. Con esta ruta se obtiene una excepción durante la ejecución del proceso al encontrar que el "date_signature" introducido en un JSON corresponde a un paciente que ya ha sido vacunado.
2) test2_not_ok: Recorre los nodos en el siguiente orden: 1, 3, 5, 7, 9, 10, 12, 13, 15, 17, 19, 20, 21, 22. Con esta ruta se obtiene una excepción al detectar que la "date_signature" introducida en el JSON no tiene asignada una cita.
3) test3_not_ok: Recorre los nodos en el siguiente orden: 1, 3, 5, 7, 9, 10, 12, 13, 15, 17, 19, 20, 21, 23, 24. Con esta ruta se obtiene una excepción que indica que la cita asignada al "date_signature" introducido ya ha sido cancelada previamente (atributo isCancel = "Final").
4) test4_not_ok: Recorre los nodos en el siguiente orden: 1, 3, 5, 7, 9, 10, 12, 13, 15, 17, 19, 20, 21, 23, 25, 26. Con esta ruta se obtiene una excepción sobre que la fecha de la cita ya ha pasado. Esto se debe a que el paciente en cuestión tiene una cita asignada, cuya fecha es anterior a la actual.
5) test5_ok: Recorre los nodos en el siguiente orden: 1, 3, 5, 7, 9, 10, 12, 13, 15, 17, 19, 20, 21, 23, 25, 27, 31, 32, 33, 34. Con esta ruta no se obtiene ningún tipo de excepción, pero al ser de tipo "Temporal" el "cancelation_type" recibido, al

final del proceso no se añadirá la cancelación en el JSON correspondiente a las cancelaciones.

6) test6_ok: Recorre los nodos en el siguiente orden: 1, 3, 5, 7, 9, 10, 12, 13, 15, 17, 19, 20, 21, 23, 25, 27, 28, 29, 30, 31, 32, 33, 34. Con esta ruta se alcanza el mismo final que con el anterior caso de prueba. Sin embargo, al ser de tipo "Final" el "cancelation_type", la cancelación se añade al JSON correspondiente a las cancelaciones.
7) test_cancel_appointment_no_ok_parameter2: en este test se lleva a cabo un bucle for para generar subtests de todas las posibles comprobaciones necesarias para todos los casos planteados en los JSON (basados en los nodos del árbol de derivación).
8) test7_not_ok_vl_reason: Recorre los nodos en el siguiente orden: 1, 3, 5, 7, 9, 10, 12, 13, 15, 16. Con esta ruta se obtiene una excepción que muestra el mensaje "reason is nor valid", pues se detecta al recibir un valor de "reason" (str) que no corresponde con la longitud mínima. En este caso se ha introducido una descripción que posee un carácter de longitud, no alcanzando el mínimo, que es dos.
9) test8_not_ok_vl_reason: El recorrido de los nodos del diagrama en este test es el mismo que para el anterior. Por lo tanto, con éste también se obtiene la misma excepción. Pues se ha introducido un string que no cumple con la longitud debida. En este caso, en lugar de ser un único carácter, son 101.
10) test9_ok_vl_reason: El recorrido de los nodos es el siguiente y se debe a que el valor de "cancelation_type" es "Temporal" y el resto de valores introducidos son válidos: 1, 3, 5, 7, 9, 10, 12, 13, 15, 17, 19, 20, 21, 23, 25, 27, 31, 32, 33, 34. Con esta ruta no se obtiene ningún tipo de excepción. Se trata de un test que verifica que el proceso se ha realizado correctamente recibiendo un valor de "reason" con una longitud de 3 caracteres (uno más que el límite inferior establecido).
11) test10_ok_vl_reason: El recorrido de los nodos es el mismo que para el anterior, pues tiene las mismas características. Únicamente se distingue en que el valor recibido para "reason" posee una longitud de dos caracteres (también válida, ya que coincide con el límite inferior establecido).
12) test11_ok_vl_reason: El recorrido de los nodos es en el siguiente orden: 1, 3, 5, 7, 9, 10, 12, 13, 15, 17, 19, 20, 21, 23, 25, 27, 28, 29, 30, 31, 32, 33, 34. Este camino permite que la cancelación se registre correctamente gracias a que todos los

valores introducidos son válidos. El valor de "cancelation_type" es "Final" y esa es la razón por la que recorre algún nodo más que el caso de prueba previo. Además, lo importante de este test, es que el valor introducido para "reason" posee una longitud de cien caracteres, coincidiendo así con el límite superior y siendo válido para su creación.

13) test12_ok_vl_reason: Recorre el mismo camino que los tests 9 y 10, ya que contiene las mismas propiedades. No obstante, este caso de prueba ha sido diseñado para validar que introduciendo un string de 99 caracteres como valor de "reason", el proceso se lleva a cabo sin problemas y el registro de la cancelación se completa exitosamente.
14) test13_ok: Recorre los nodos en el siguiente orden: 1, 3, 5, 7, 9, 10, 12, 13, 15, 17, 19, 20, 21, 23, 25, 27, 28, 29, 30, 31, 32, 33, 34. . Con este test se obtiene un registro de la cancelación exitosa a pesar de llevar a cabo dos cancelaciones de una misma cita. Esto se debe a que en primer lugar se realiza una cancelación de tipo temporal; y en segundo lugar, una cancelación de tipo final.
15) test14_not_ok: Recorre los nodos en el siguiente orden: 1, 3, 5, 7, 9, 10, 12, 13, 15, 17, 19, 20, 21, 23, 24. Con esta ruta se obtiene una excepción que muestra el mensaje "La cita ya ha sido cancelada anteriormente". Esto se debe a que en el test se ha creado una nueva cita para un paciente determinado y se han tratado de hacer dos cancelaciones de dicha cita. En primer lugar, la cancelación es de tipo "Final" y la segunda es de tipo "Temporal". En la base termina quedando registrada únicamente la primera cancelación.

Entre todos estos tests se puede realizar una clasificación en función de la técnica empleada para llevarlos a cabo. Los tests relacionados con **valores límite** serían los tests 7, 8, 9, 10, 11 y 12, ya que se evalúa la posibilidad de éxito o error en caso de recibir strings de diferente longitud como valor de "reason" en el JSON de cancelación. Por otro lado, todos los tests restantes se consideran como parte del grupo de la técnica conocida como **clases de equivalencia.**

2.2.3 Gramática

A continuación, se muestra la gramática planteada para la correcta creación de los casos de pruebas y la generación del árbol que se expone en el siguiente apartado.

G = ({ I, Inicio, DS, Coma, CT, Re, Final, AtribDS, Puntos, ValorDS, AtribCT, ValorCT, AtribRe, ValorRe, Comillas, HexID, Type, Desc}, { ^([0-9][A-F]){64}\Z , { , } , , , “ , : , ^[Temporal|Final]\Z, ^{2-100}\Z, Date_signature, Cancelation_type, Reason}, P, I)

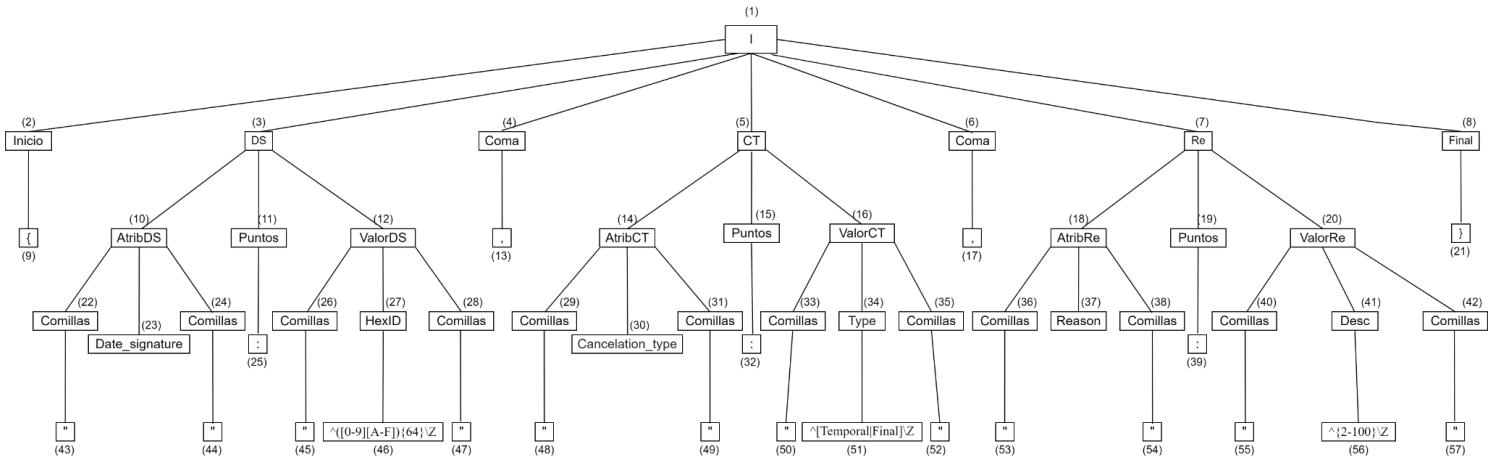
P = {

I ::= Inicio DS Coma CT Coma Re Final
DS ::= AtribDS Puntos ValorDS
AtribDS ::= Comillas Date_signature Comillas
ValorDS ::= Comillas HexID Comillas
HexID ::= ^([0-9][A-F]){64}\Z
CT ::= AtribCT Puntos ValorCT
AtribCT ::= Comillas Cancelation_type Comillas
ValorCT ::= Comillas Type Comillas
Type ::= ^[Temporal|Final]\Z
Re ::= AtribRe Puntos ValorRe
AtribRe ::= Comillas Reason Comillas
ValorRe ::= Comillas Desc Comillas
Desc ::= ^{2-100}\Z
Inicio ::= {
Final ::= }
Coma ::= ,
Comillas ::= “
Puntos ::= :

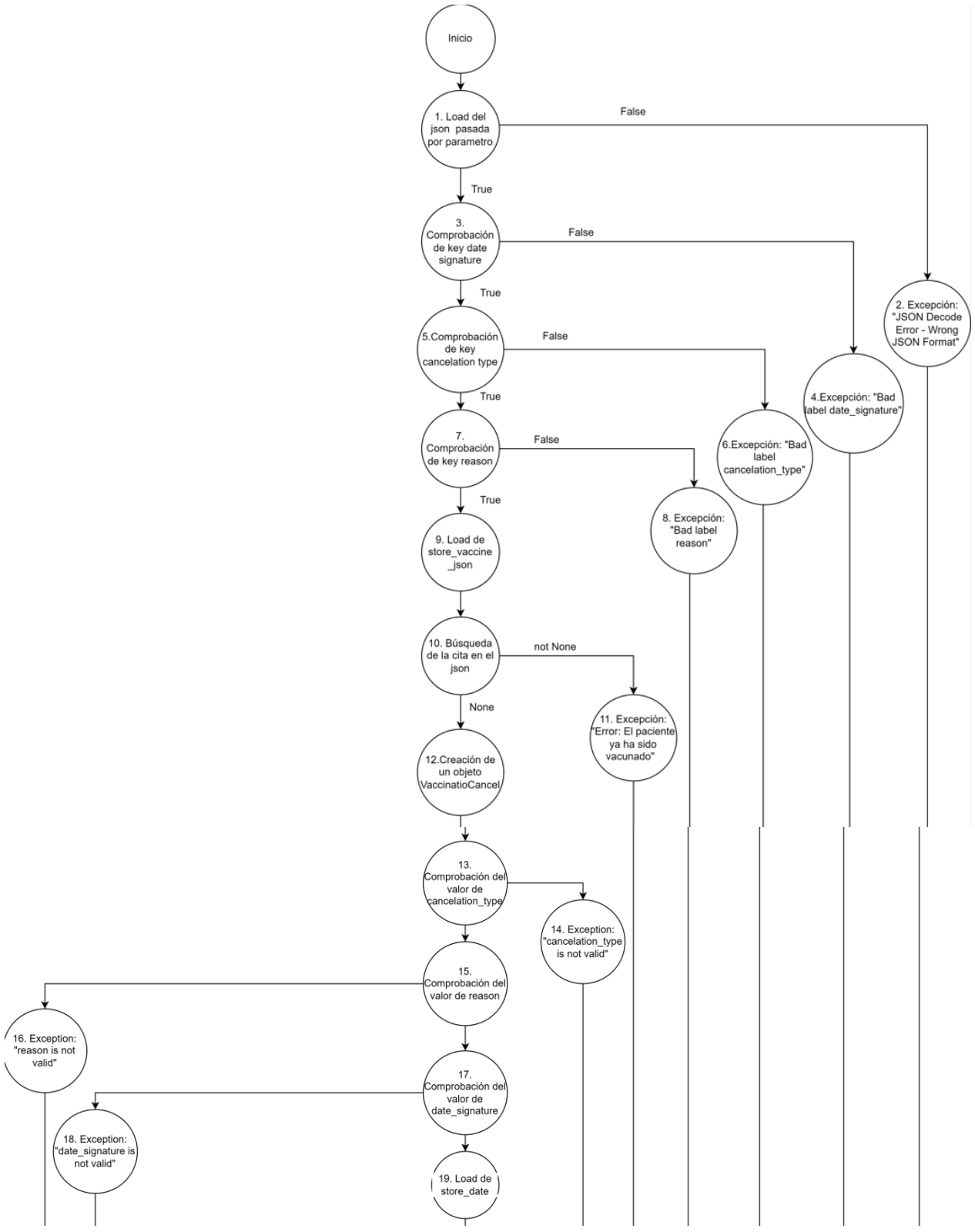
}

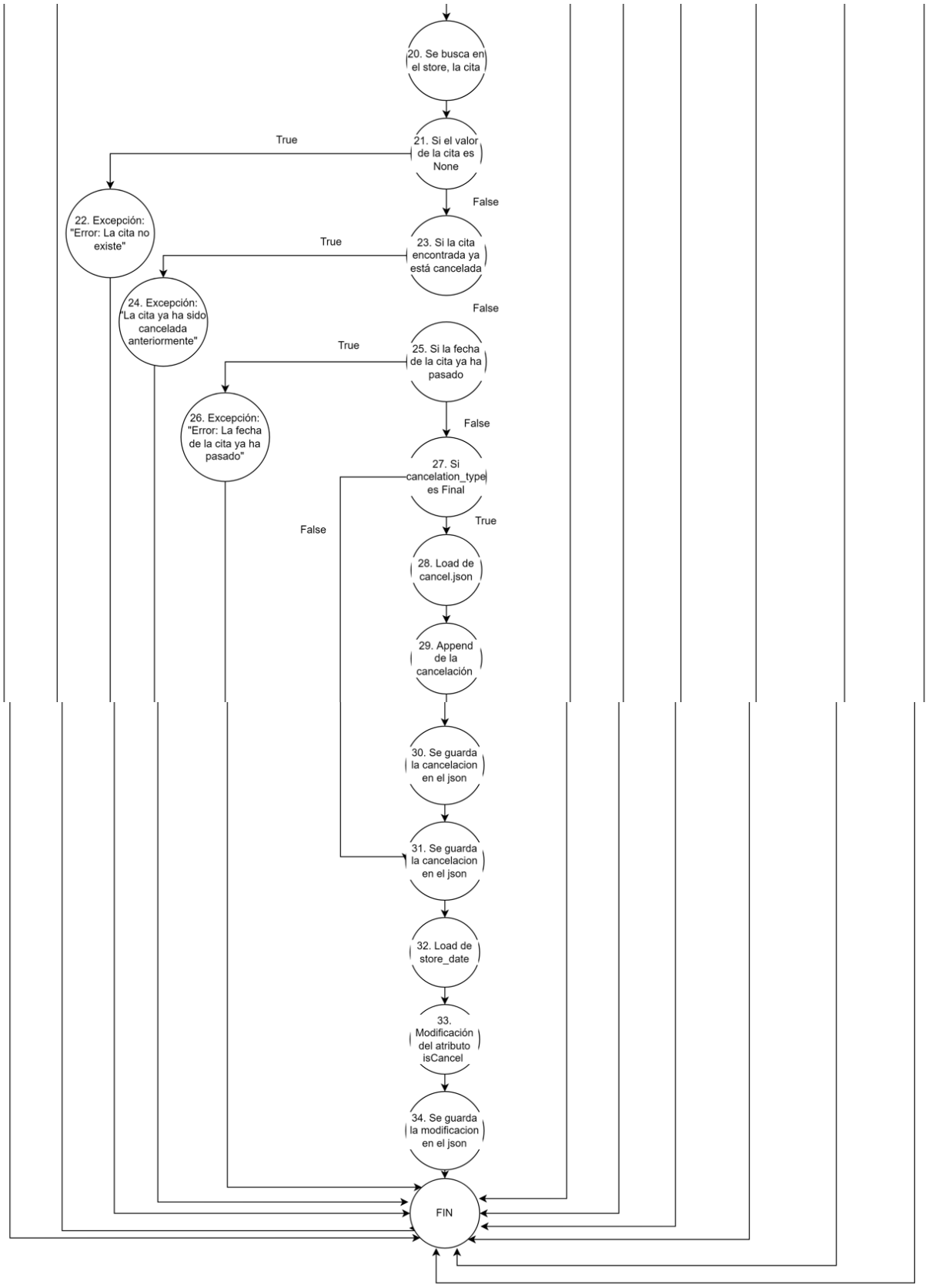
2.2.4 Árbol de derivación

A continuación, se expone el árbol de derivación que hemos generado en base a la gramática planteada para la correcta resolución de la práctica.
Al igual que en la práctica EG3, cada uno de los nodos que aparecen han sido enumerados en un orden lógico. En nuestro caso, de arriba hacia abajo y de izquierda a derecha.
En el árbol se pueden distinguir dos tipos diferentes de nodos: los **no terminales** (como el 3, 4 o 5) y los **terminales** (como el 43, 44 o 45). Para los nodos no terminales se han generado en el Excel y en código, únicamente tests relacionados con la duplicación y eliminación de los mismos. En cuanto a los terminales, se han creado tests de modificación junto con los anteriores.
No obstante, no todos los nodos terminales contienen los tres tipos de tests, ya que hay ciertos nodos que son subárboles que se repiten. Es por ello, que al igual que en el anterior trabajo, consideramos innecesario realizar un mismo caso de prueba.



2.2.5 Gráfico de flujo de control




20. Se busca en el store, la cita
21. Si el valor de la cita es None
True
22. Excepción: "Error: La cita no existe"
False
23. Si la cita encontrada ya está cancelada
True
24. Excepción: "La cita ya ha sido cancelada anteriormente"
False
25. Si la fecha de la cita ya ha pasado
True
26. Excepción: "Error: La fecha de la cita ya ha pasado"
False
27. Si cancelation_type es Final
True
False
28. Load de cancel.json
29. Append de la cancelación
30. Se guarda la cancelacion en el json
31. Se guarda la cancelacion en el json
32. Load de store_date
33. Modificación del atributo isCancel
34. Se guarda la modificacion en el json
FIN

3. CONCLUSIONES

Con el desarrollo de esta última práctica, hemos podido consolidar todos los conocimientos, habilidades y dotes que hemos adquirido a lo largo del curso para poder alcanzar los objetivos pedidos. Todo ello nos ha ayudado a mejorar considerablemente a lo largo de la realización del trabajo. Esta práctica ha sido laboriosa, ya que se pedía realizar un gran número de cosas. Sin embargo, nos ha resultado algo más sencilla que prácticas anteriores. Por último, queremos destacar la experiencia que esta práctica y asignatura nos ha ofrecido, ya que nos ha permitido la oportunidad de mejorar en el ámbito de la programación y en aspectos relacionados con la cooperación y el trabajo en grupo. Además, queremos agradecer la ayuda que nos han presentado los vídeos propuestos en el aula global y las dudas resueltas por la profesora. Gracias a dicho apoyo y a la reiteración de intentos que hemos llevado a cabo, hemos podido resolver todos los problemas que nos iban surgiendo.