Desarrollo de software - Práctica 3

Diego Del Pozo Fernández - 100451056
Carlos Saiz Blanco - 100451312

Índice

Casos de prueba:

Primera función: request vaccination id

Para la primera función hemos realizado una gran variedad de tests, puesto que hay muchos más casos disponibles para comprobar que en las otras dos funciones, incluyo debajo una breve definición de cada uno.

Los dos primeros test, se centran en comprobar (tras crear el objeto vaccinemanager), el caso en el que la uuid sea incorrecta (saltando una excepción), y el caso en el que la uuid sea correcta devolviendo True.

Los siguientes 4 tests de edad, comprueban los casos en los que la edad introducida sea, respectivamente inferior al límite permitido, superior al límite, un str, y un float (que al no ser int daría también una excepción), obteniendo así la excepción Invalid age en estos casos incorrectos.

Posteriormente tenemos 4 test que, al igual que la edad, comprueban el nombre en los casos de, que esté vacío, sea demasiado largo, no tenga un apellido, y no sea un string (por ejemplo un int), dando las excepciones esperadas para el nombre.

Después tenemos tres tests que comprueban el patient id, el primero comprueba un patient id que no es un string (dando excepción), el siguiente comprueba un patient id incorrecto, (dando excepción también), y el último realiza un registro correcto del patient id, comprobando que se ha realizado correctamente.

Continuamos con diversos test que, siguiendo el mismo método que en edad y nombre, comprueban algunos casos incorrectos para el número de teléfono, y otros test que comprueban dos casos incorrectos para el tipo de registro.

Por último tenemos un test que comprueba el guardado correcto del json, (con sus datos dentro), y varios tests que comprueban el hash devuelto por la función, tanto su formato (32 dígitos), como si es correcto.

Casos de prueba:

Segunda función: get vaccine date

La gramática que hemos propuesto es la siguiente:

FICHERO ::= INICIO DATOS FIN
INICIO ::= {

FIN ::= }

DATOS ::= CAMPO1 SEPARADOR CAMPO2

CAMPO1 ::= CLAVE1 IGUALDAD DATO1

CAMPO2 ::= CLAVE2 IGUALDAD DATO2

IGUALDAD ::= :

SEPARADOR ::= ,

CLAVE1 ::= COMILLAS VALOR_CLAVE1 COMILLAS

CLAVE2 ::= COMILLAS VALOR_CLAVE2 COMILLAS

VALOR_CLAVE1 ::= PatientSystemID

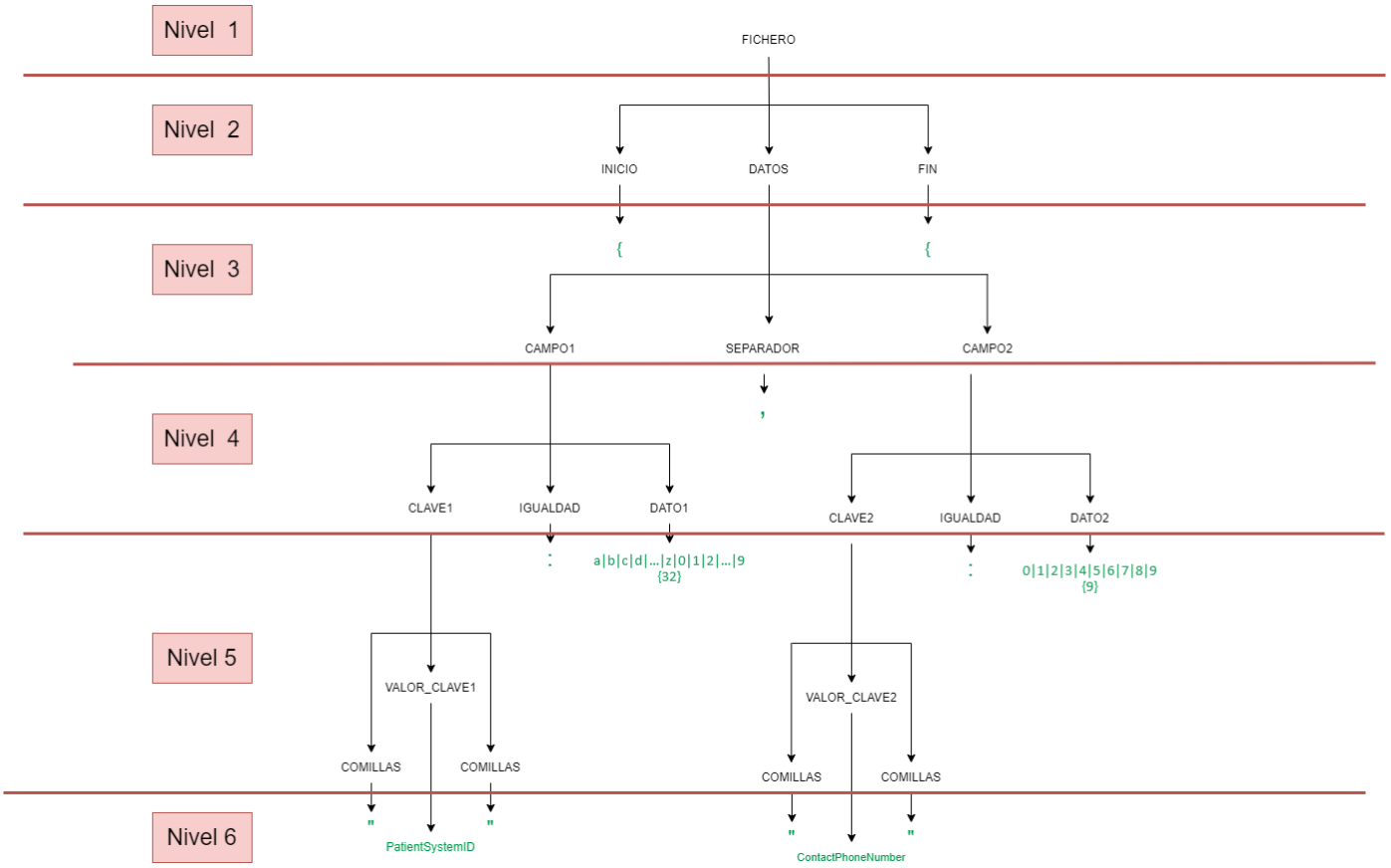
VALOR_CLAVE2 ::= ContactPhoneNumber

DATO1 ::= a|b|c|d|…|z|0|1|2|…|9 {32}

COMILLAS ::= “”

DATO2::= 0|1|2|3|4|5|6|7|8|9 {9}

El árbol de derivación de la gramática es el siguiente:

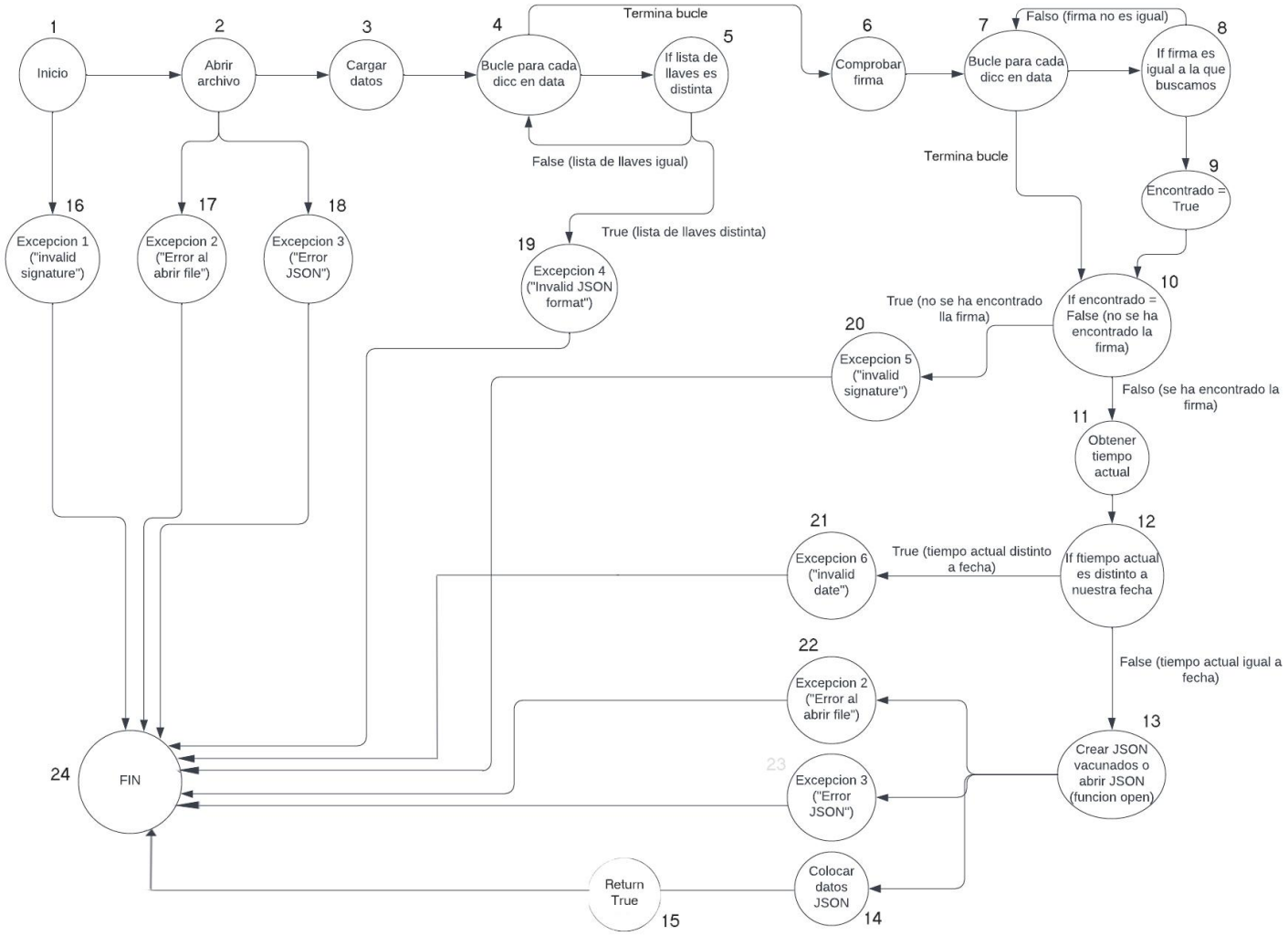


El árbol de derivación tiene todas sus hojas como nodos terminales (marcados en verde) y tiene 6 niveles distintos. Está hecho a partir de una gramática de tipo 2 y muestra las combinaciones válidas para el JSON que debe recibir la función 2

Los test comprueban que todos los campos son correctos, empezando por comprobar que el JSON existe (test_json_no_existe()), que el archivo es un JSON (test_no_es_json()) y posteriormente se verifica que la función devuelve error en caso de introducir un caso no válido en la gramática, además de verificar los tipos de los parámetros y los valores límite.

Casos de prueba:

Tercera función: vaccine patient



A la hora de realizar los casos de prueba de esta función, se ha tenido en cuenta el diagrama de flujo insertado arriba, realizando una comprobación de las excepciones y probando caminos distintos hasta llegar al mismo punto final.

|

|

V

Los primeros *test_firma_none y test_firma_int* comprueban, respectivamente, que ocurriría en el caso de que la firma que se diese como argumento fuese none, y que ocurriría si la firma fuese del tipo int, dando como resultado la excepción esperada invalid signature.

Posteriormente *(test_firma_63caracteres)* también se comprueba esa excepción desde otra perspectiva, en este caso la firma es de 63 caracteres en vez de 64, por lo que comprueba también la excepción.

Tras haber hecho esas comprobaciones, pasamos a los dos siguientes test *test_comprobar_json_registered_vaccinations y test_comprobar_json_appoinments,* que comprueban la existencia de los jsons mencionados.

Luego tenemos otro *test_no_encontrada_firma* que comprueba con una firma incorrecta (una que no esté en el documento), obteniendo así otra de las excepciones.

Por último, realizamos un test correcto *test_funcion_correcta* (que compruebe la devolución de True), y otr*o test_comprobar_formato_JSON* que comprueba el formato del json registered vaccinations.

Rutas del diagrama y explicación de casos con los bucles:

-La función recorre desde el nodo 1 hasta el nodo 4, en donde hay un bucle que pasa al nodo 5, en donde comprueba constantemente la condición (hasta terminar el bucle), volviendo al nodo 4, y otra vez al 5 a comprobar.
Si termina el bucle, quiere decir que no hay ninguna distinta, por lo que sería correcto, saltando al bucle 6, en donde recorre hasta el nodo 7 (otro bucle) que tiene la condición en el nodo 8.
En el caso de que se encuentre la firma en el nodo 8, se tomará un camino hacia el nodo 9 y 10, similar al camino que toma cuando no se encuentra.
Sin embargo, aquí comprueba si se ha encontrado o no, llevando a una excepción en el caso de que no se haya encontrado.
Si se ha encontrado sigue el camino hasta 14 y el 15 (devuelve True) llegando al final (nodo 24).

-La función también puede desviarse por alguna excepcion del principio, ya sean los nodos 16,17,18,19 (el nodo 19 es en el caso de que la función encuentre una llave en el diccionario distinta), o por alguna de las excepciones finales 20,21,22,23, llevando todas las excepciones directamente al nodo final.