Software Development

BACHELOR IN COMPUTER SCIENCE AND ENGINEERING

Final Project

Isaac Buiza González

(100451176, group 89, 100451176@alumnos.uc3m.es)

Valentina Cataldo Jiménez

(100451232, group 89, 100451232@alumnos.uc3m.es)

uc3m

uc3m

1. Get_vaccine_date

For the function *get_vaccine_date* we have created a python file for the attribute *appointment_date* and we edit the function *create_appointment_from_json_file*. For making the function to check the date as iso format, on the python file for the *appointment_date* we add a regex function and compare it with “today” also in iso format. There are some checking to ensure that the date entered is correct

a. Equivalence classes

In the equivalence classes, we check that the format of the year, month and day are strings and they have 4, 2 and 2 digits, respectively. Also, we check that the separators between each one it is with ‘-’ and no other type of separator.

b. Boundary values

In the boundary values, we check that the date the user asks for is higher than the current date and that the month and day are not out of range. For the month from 1 to 12 and for the day from 1 to 30.

2. Cancel_appointment

For the function cancel_appointment we created an instance of an object from the data received in the json file as an input in order to check the correctness of the fields. Once the instance has been created a series of checking are done to ensure the coherence of the data in the system (not canceling a past appointment, not canceling an already canceled appointment, etc)

a. Equivalence classes

In the equivalence classes, we check that the format of the date signature, cancellation type and reason are strings and that the cancellation type is equal to “Final” or “Temporal”.

b. Boundary values

Applied to the fields of.

Date_signature: expected length of 64 characters, so files with lengths of 63,64 and 65 were checked.

Reason: Expected length between 2 and 100 characters, so files with lengths of 1,2,3,99,100 and 101 characters were checked.

c. Nodes

We defined the following grammar for the input json files:

File::= Begin_Object Data End_object
Begin_Object ::= {
End_object ::= }
Data ::= Field1 Separator Field2 Separator Field3
Field1 ::= Label_Field1 Equals Value_Field1
Field2 ::= Label_Field2 Equals Value_Field2
Field3 ::= Label_Field3 Equals Value_Field3
Separator ::= ,
Equals ::= :
Quotation ::= "

Label_Field1 ::= Quotation Value_Label1 Quotation
Value_label1 ::= date_signature
Value_Field1 ::= Quotation Value1 Quotation
Value1 ::= 0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F (64)

Label_Field2 ::= Quotation Value_Label2 Quotation
Value_label2 ::= cancelation_type
Value_Field2 ::= Quotation Value2 Quotation
Value2 ::= Temporal | Final

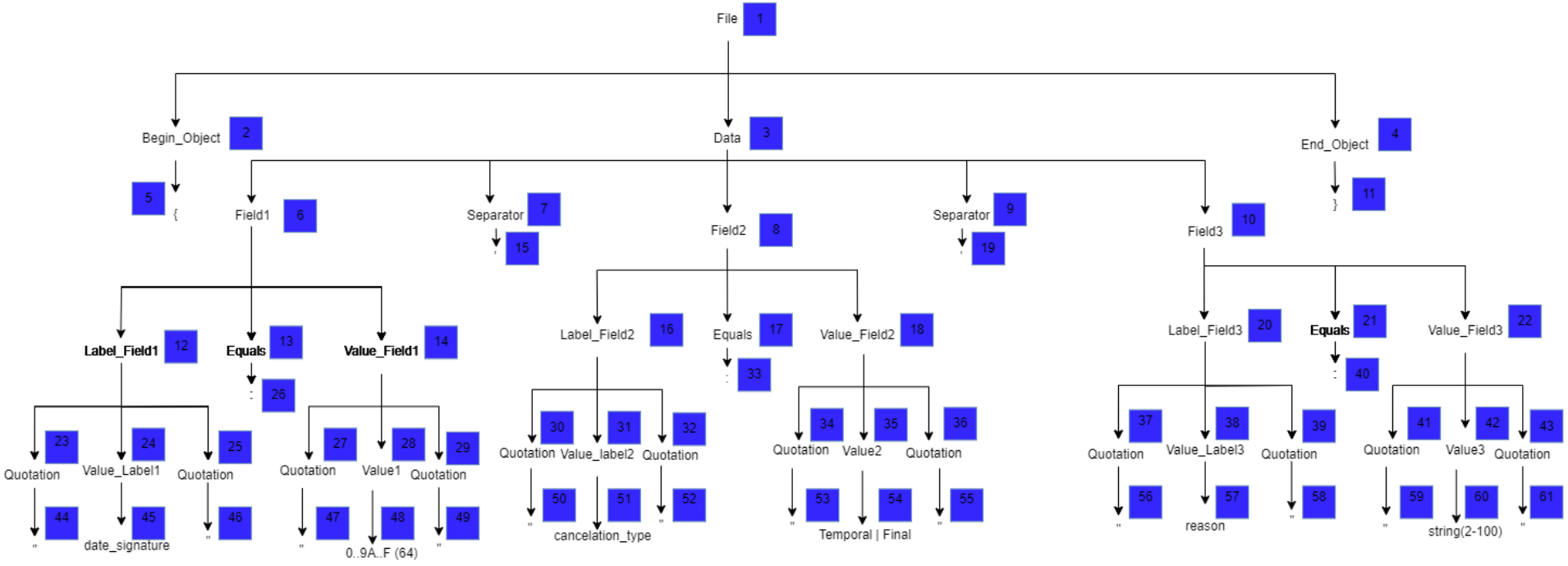
Label_Field3 ::= Quotation Value_Label3 Quotation
Value_label3 ::= reason
Value_Field3 ::= Quotation Value3 Quotation
Value3 ::= string(2-100)

Computing its derivation tree we got 61 nodes which were tested using the tests defined in the excel file.
(Also see diagram at the end of the document)

d. Other tests

Apart from the test that have already been explained we decided to implement 4 additional tests that ensure the correctness of the data in the system this tests are :
(test_appointment_does_not_exist, test_appointment_already_passed, test_vaccine_already_administered, test_vaccine_already_canceled)

The implementation of these tests ensures that an inexistent appointment cant be deleted, a past appointment cant be deleted, an already administered appointment cant be deleted and that any appointment cannot be canceled twice.


File 1
Begin_Object 2
Data 3
End_Object 4
5 {
Field1 6
Separator 7
15 ,
Field2 8
Separator 9
19 ,
Field3 10
11 }
Label_Field1 12
Equals 13
Value_Field1 14
26 :
23 Quotation
24 Value_Label1
25 Quotation
44 "
45 date_signature
46 "
27 Quotation
28 Value1
29 Quotation
47 "
48 0..9A..F (64)
49 "
Label_Field2 16
Equals 17
Value_Field2 18
33 :
30 Quotation
31 Value_label2
32 Quotation
50 "
51 cancelation_type
52 "
34 Quotation
35 Value2
36 Quotation
53 "
54 Temporal | Final
55 "
Label_Field3 20
Equals 21
Value_Field3 22
40 :
37 Quotation
38 Value_Label3
39 Quotation
56 "
57 reason
58 "
41 Quotation
42 Value3
43 Quotation
59 "
60 string(2-100)
61 "

3. Conclusions

Developing this project we noticed that developing functions that rely heavily on dates can cause a lot of problems during the testing phase of the project. For this, the existence of the freezetime function in python mitigates this problem but it is still a hustle to deal with.
Also we started to value the existence of online repositories such as github since they make it quite easy to follow the pair programming methodologies and are extremely useful to develop a group project even when the other member does not have a lot of free time at the same time.
And last but not least even though this was a relatively large project it was fulfilling to complete as it is somewhat similar to what we expect to find in our professional environment.