EJERCICIO GUIADO 3
DESARROLLO DE SOFTWARE

Alberto Blasco Ventas - 100429068-80-100429068@alumnos.uc3m.es
Fidel Navacerrada García-100429012-80-100429012@alumnos.uc3m.es

ÍNDICE

INTRODUCCIÓN

En este pdf, hemos explicado de forma breve lo que hace cada función de la práctica 3 de Desarrollo de Software, además de incluir una serie de casos de prueba para cada función en los que se puede encontrar de forma rápida el resultado de los test que hemos realizado.

En el Pycharm todo el código importante lo hemos desarrollado en en **access_manager.py**.

- En **“test_request_access_code_test.py”** desarrollamos los test de la primera función.
- En **“test_get_access_key_test.py”** desarrollamos los test de la segunda función.
- En **“test_access_manager_test.py”** desarrollamos los test de la tercera función.

Terminamos el documento con una opinión sobre la práctica y las mayores dificultades que hemos tenido que superar respecto a esta.

FUNCIÓN 1 (request_access_code(...))

La primera función de la práctica, nos pide que la persona que quiera solicitar una llave (**“key”**) tenga antes que pasar sus datos personales como parámetros para que la función le devuelva un código (**clave de acceso**) para posteriormente poder solicitar la clave de la llave para poder abrir las puertas.

Además la función debe ser capaz de validar que los datos incluidos en los parámetros sean correctos, no vale meter un DNI con espacios o con dos letras, etc. Por último la función debe ser capaz de crear un fichero (**storeRequest.json**) en el que irá introduciendo las solicitudes hechas por las personas que quieren recibir su código de acceso y si se da el caso en el que un DNI está ya introducido en el fichero (**storeRequest.json**) no lo introducirá e indicará que ese DNI ya se encuentra en el fichero.

CASOS DE PRUEBA

TÉCNICA	ID_TEST	DESCRIPCIÓN
Valores Límites	test_request_access_code_dni_guest_ok_1	La función funciona correctamente 1
Valores Límites	test_request_access_code_dni_resident_ok_2	La función funciona correctamente 2
Clases de Equivalencia	test_request_access_code_dni_repetido_3	Caso qué ocurre si se da la situación
Valores Límites	test_request_access_code_7_numeros_4	Cuando el DNI tiene 7 números
Valores Límites	test_request_access_code_8_numeros_5	Cuando el DNI tiene 9 numeros
Valores Límites	test_request_access_code_no_letra_dni_6	Cuando el DNI no tiene una letra
Valores Límites	test_request_access_code_2_letras_dni_7	Cuando el DNI tiene dos letras
Clases de Equivalencia	test_request_access_code_letra_numeros_cambiados_8	Cuando el DNI empieza por la letra en vez de números
Clases de Equivalencia	test_request_access_code_letra_entre_numeros_9	La letra se inserta en medio del número en vez de al final
Clases de Equivalencia	test_request_access_code_tipo_de_acceso_distinto_10	El tipo de acceso es diferente a Guest y Resident
Valores Límites	test_request_access_code_guest_menos_dias_11	Guest tiene menos días de los necesarios
Valores Límites	test_request_access_code_guest_mas_dias_12	Guest tiene más días de los necesarios
Clases de Equivalencia	test_request_access_code_resident_num_dist_0_13	Guest tiene otro día distinto de 0 en el número de días
Clases de Equivalencia	test_request_access_code_nums_end_email_14	El email contiene números en el dominio
Clases de Equivalencia	test_request_access_code_no	No hay @ en el email

	_at_email_15	
Clases de Equivalencia	test_request_access_code_blank_email_16	El email está en blanco
Clases de Equivalencia	test_request_access_code_no_domain_email_17	No existe dominio en el email
Clases de Equivalencia	test_request_access_code_no_domain_2_email_18	Entre el dominio y el @ no hay nada en el email
Clases de Equivalencia	test_request_access_code_no_email_19	El email está en blanco
Clases de Equivalencia	test_request_access_code_no_dni_20	El DNI está en blanco
Clases de Equivalencia	test_request_access_code_no_name_21	El nombre está en blanco
Clases de Equivalencia	test_request_access_code_int_name_22	El nombre contiene números
Clases de Equivalencia	test_request_access_code_name_no_surname_23	Solamente hay nombre, no apellidos

FUNCIÓN 2 (get_access_key(...))

La segunda función del programa recibe como parámetro la ruta de acceso a un fichero, este fichero tendrá un diccionario con un código (**access_request**), el dni y los emails.
La función lo que va ha hacer es abrir tanto el fichero pasado por parámetros como el fichero creado en la función 1 con todas las solicitudes, después de eso va a comprobar que el **DNI** y el **Access_Request** del fichero pasado por parámetro están en el fichero de la función 1.

Si se cumple la condición creamos un objeto de la clase **Access_Key(),** y por último se crea un nuevo fichero que va a almacenar todos los atributos del objeto de tipo key creado anteriormente, pero además añadiendo el código de acceso(**access_request**), para que lo almacene también.

La función devuelve el parámetro “key” del objeto key, y ese parámetro es un código que necesitará el usuario para abrir las pruebas de los edificios y tendrá una fecha de expiración.

CASOS DE PRUEBA

NODO	IDENTIFICADOR	DESCRIPCIÓN	FICHERO JSON
1	T_1	correcto con 2 email	correcto_2_emails.json
1	T_2	correcto con 1 email	correcto_1_email.json
1	T_3	fichero con otro formato	no_es_json
3	T_4	los datos no se encuentran en storeRequest	no_data_in_storeRequest.json
2	T_5	2 brackets de comienzo	double_start_bracket.json
2	T_6	no hay bracket de comienzo	no_start_bracket.json
4	T_7	no hay bracket de final	no_end_bracket.json
4	T_8	doble bracket de final	double_end_bracket.json
3	T_9	no hay datos	no_datos.json
3	T_10	datos duplicados	datos_repetidos.json
6	T_11	campo 1 repetido	campo_1_repetido.json
6	T_12	no hay campo 1	no_campo_1.json
7	T_13	separador 1 duplicado	separador_1_repetido.json
7	T_14	sin separador 1	sin_separador_1.json

8	T_15	campo 2 duplicado	campo_2_repetido.json
8	T_16	sin campo 2	sin_campo_2.json
9	T_17	sin separador 2	sin_separador_2.json
9	T_18	separador 2 duplicado	separador_2_repetido.json
10	T_19	sin campo 3	sin_campo_3.json
10	T_20	campo 3 duplicado	campo_3_repetido.json
11	T_21	etiqueta 1 duplicada	e1_repetida.json
11	T_22	no hay etiqueta 1	sin_e1.json
12	T_23	no hay igualdad 1	sin_igualdad1.json
12	T_24	igualdad 1 duplicada	doble_igualdad1.json
13	T_25	valor dato 1 duplicado	doble_valor_dato1.json
13	T_26	sin valor dato 1	sin_valor_dato1.json
22	T_27	sin comillas etiqueta 1	sin_comillas_e1.json
22	T_28	comillas etiqueta 1 duplicadas	doble_comillas_e1.json
23	T_29	accesscode duplicado	doble_AccessCode.json
23	T_30	sin accesscode	sin_AccessCode.json
26	T_31	sin comillas en valor dato 1	sin_comillas_VD1.json
26	T_32	comillas vd1 duplicadas	dobles_comillas_VD1.json
27	T_33	valor 1 duplicado	doble_v1.json
27	T_34	sin valor 1	sin_v1.json
15	T_35	sin etiqueta 2	sin_etiqueta_2.json
15	T_36	etiqueta 2 duplicada	doble_etiqueta_2.json
16	T_37	no hay igualador 2	sin_igualador_2.json
16	T_38	igualador 2 duplicado	doble_igualador_2.json
17	T_39	sin valor dato 2	sin_valor_dato_2.json
17	T_40	valor dato 2 duplicado	doble_valor_dato_2.json

29	T_41	sin comillas etiqueta 2	sin_comillas_e2.json
29	T_42	comillas etiqueta 2 duplicadas	doble_comillas_e2.json
30	T_43	dni duplicado	doble_DNI.json
30	T_44	sin dni	sin_DNI.json
33	T_45	sin comillas valor dato 2	sin_comillas_vd2.json
33	T_46	comillas valor dato 2 duplicadas	dobles_comillas_vd2.json
34	T_47	dato dni duplicado	doble_dato_DNI.json
34	T_48	sin dato dni	sin_dato_DNI.json
54	T_49	sin letra en dni	sin_letra_dni.json
54	T_50	letra dni duplicada	doble_letra_dni.json
53	T_51	números dni duplicados	doble_numeros_dni.json
53	T_52	sin números dni	sin_numeros_dni.json
19	T_53	sin etiqueta 3	sin_etiqueta_3.json
19	T_54	etiqueta 3 duplicada	doble_etiqueta_3.json
20	T_55	sin igualador 3	sin_igualador_3.json
20	T_56	igualador 3 duplicado	doble_igualador_3.json
21	T_57	valor dato 3 duplicado	doble_vd3.json
21	T_58	sin valor dato 3	sin_vd3.json
36	T_59	sin comillas etiqueta 3	sin_comillas_e3.json
36	T_60	comillas etiqueta 3 duplicadas	doble_comillas_e3.json
37	T_61	notificationmail duplicado	doble_NotificationMail.json
37	T_62	sin notificationmail	sin_NotificationMail.json

40	T_63	sin corchetes dato 3	sin_corchs_dato3.json
40	T_64	corchetes dato3 duplicados	dobles_corchs_dato3.json
60	T_65	sin comillas dato3	sin_comillas_dato3.json
61	T_66	sin valor3	sin_v3.json
61	T_67	valor3 duplicado	doble_v3.json

GRAMÁTICA

Fichero::= <Inicio objeto><Datos> <Fin objeto>
Inicio objeto ::= {
Fin objeto ::= }
Datos::= <Campo1>< Separador><Campo2><Separador><Campo3>
Campo 1::= <Etiqueta 1><Igualdad><Valor dato 1>
Etiqueta 1::=<Comilla><Valor Etiqueta 1><Comilla>
Valor Etiqueta 1 ::= AccessCode
Valor dato 1 ::= <Comilla><Valor 1><Comilla>
Valor 1 ::= a | b | c | d | e | f | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 (32)

Campo 2::= <Etiqueta 2><Igualdad><Valor dato 2>
Etiqueta 2::=<Comilla><Valor Etiqueta 2><Comilla>
Valor Etiqueta 2 ::= DNI
Valor dato 2 ::= <Comilla><DNI><Comilla>
DNI ::= <Números><Letra>
Números::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 (8)
Letra ::= A | B | C | D | E | F | G | H | J | K | L | M | P | Q | R | S | T | V | W | X | Y | Z (1)

Campo 3::= <Etiqueta 3><Igualdad><Valor dato 3>
Etiqueta 3::=<Comilla><Valor Etiqueta 3><Comilla>
Valor Etiqueta 3 ::= NotificationMail
Valor dato 3 ::= <Corchete inicio><dato3><Corchete fin>
dato 3 ::= <Comilla><Valor 3><Comilla>
Valor 3::= <Comillas><email><Comillas>(<separador> <comillas><email><comillas>) {0-4}
email::= a | b | c | d | e | f | g | h | i | j | k | l | m | ñ | o | p | q | r | s | t | u | v | w | x | y | z
---
Separador::= ,
Igualdad::= :
Comilla::= "

Corchete inicio::= [Corchete fin::=]

ÁRBOL DE DERIVACIÓN

Como el árbol de derivación no nos cabía en la memoria hemos agregado un documento de tipo **PDF** llamado **árbol de derivación** en el que se puede ver de manera más clara.

FUNCIÓN 3(open door(...))

La última función del proyecto se encarga de recibir por parámetros una llave. Lo primero que hará la función es verificar si el código introducido tiene el formato correcto mediante una expresión regex.

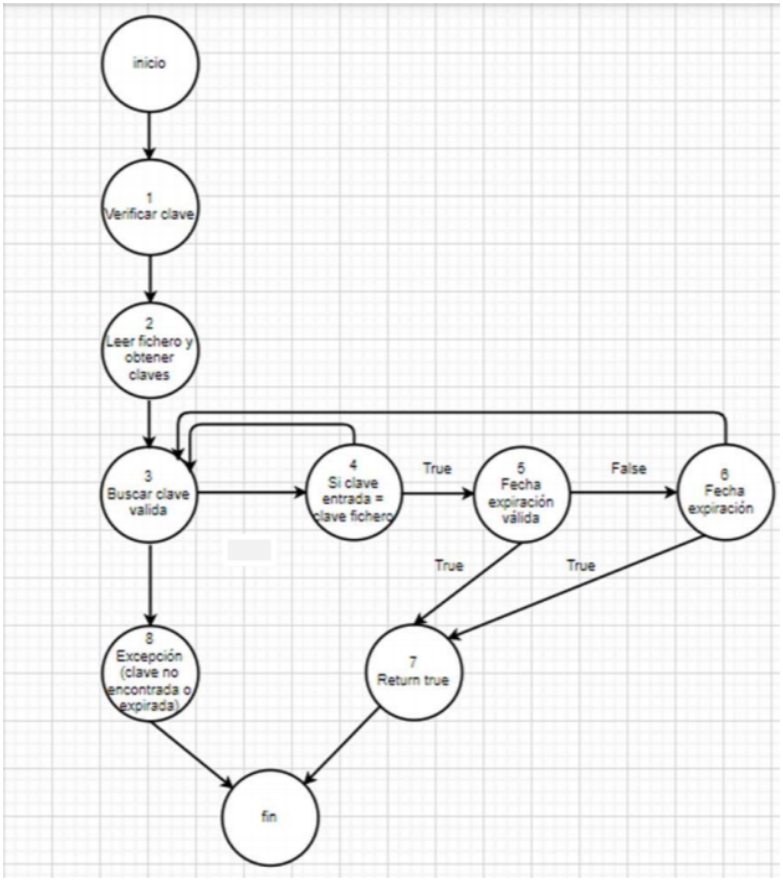
Luego abre el fichero donde están almacenadas todas las llaves y comprobamos que la llave esté introducida y que el tiempo de expiración no ha pasado, si se cumplen esas dos condiciones la función devuelve True, en caso contrario lanzará una excepción.

CASOS DE PRUEBA

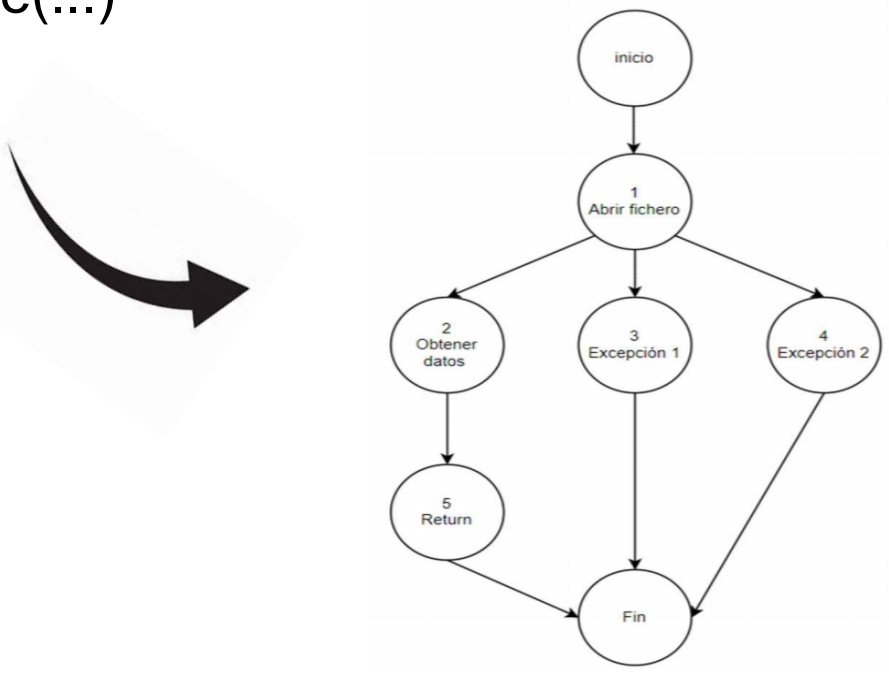
PATH	ID TEST	DESCRIPTION	EXPECTED RESULT
1_2_3_4_5_7_FIN	test_comprobar_keys_correcta_1	Test válido, recorre el camino indicado para devolver la clave, que no ha expirado.	OK
1_2_3_4_5_7_FIN	test_comprobar_keys_correcta_2	Test válido, recorre el camino indicado para devolver la clave de un residente (days==0).	OK
1_2_3_4_5_8_FIN	test_comprobar_keys_incorrectos_1	Test invalido, encuentra una clave que ya ha expirado.	key is not found or is expired
1_2_3_8_FIN	test_comprobar_keys_incorrectos_2	Test invalido, la clave introducida no tiene un formato válido.	invalid hey
Loop:3_8	loop_0times	No entra en el bucle	key is not found or is expired
Loop:3_5_6_8	loop_1times	Entra 1 vez	key is not found or is expired
Loop:3_5_6_3_5_6	loop_2times	Entra 2 veces	key is not found or is expired
Loop:3_5_6_(99)_8	loop_99times	Entra n-1 veces	key is not found or is expired
Loop:3_5_6_(100)_8	loop_100times	Entra n-1 veces	key is not found or is expired

DIAGRAMA DE FLUJO DE CONTROL

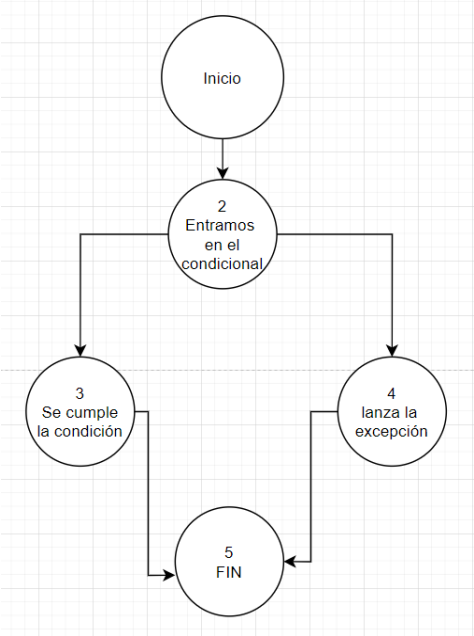
open_door(...)



read_key_file(...)



check_key(.....)



CONCLUSIÓN

En resumen, nos ha encantado hacer este trabajo porque sentimos que va a ser muy útil para nuestros futuros trabajos como desarrolladores, lo que más pesado se nos ha hecho hacer con diferencia ha sido tanto hacer los test y ficheros.joson para la función dos como pasarlo a tablas execl.