EJERCICIO GUIADO 3

DESARROLLO DIRIGIDO POR PRUEBAS

Sonsoles Molina Abad – 100432073

Lorenzo Largacha Sanz – 100432129

Doble Grado de Ingeniería Informática y ADE – Grupo 50

Desarrollo de Software

Universidad Carlos III de Madrid, Campus de Colmenarejo

18 marzo, 2022

1. Función 1: Solicitar Vacuna

Clases de equivalencia

El número de valores de entrada en todas las clases de equivalencia será uno, puesto que nuestros métodos solo se prueban con un paciente cada vez.

CE_V_01 → Se trata de una clase de equivalencia válida, dónde todas las entradas tienen como patient_id un uuid válido (string hexadecimal de 32 caracteres) con el formato correcto y en versión 4.

CE_NV_02 → Clase inválida, donde todas las entradas tienen como patient_id un uuid válido (string hexadecimal con el formato correcto), pero en versión 1.

CE_NV_03 → Clase inválida, donde todas las entradas tienen como patient_id un uuid válido (string hexadecimal con el formato correcto), pero en versión 2.

CE_NV_04 → Clase inválida, donde todas las entradas tienen como patient_id un uuid válido (string hexadecimal con el formato correcto), pero en versión 3.

CE_NV_05 → Clase inválida, donde todas las entradas tienen como patient_id un uuid válido (string hexadecimal con el formato correcto), pero en versión 5.

CE_NV_06 → Clase inválida, donde todas las entradas tienen como patient_id un uuid inválido ya que no se encuentra en formato hexadecimal o no tiene 32 caracteres.

CE_V_07 → Se trata de una clase válida, donde todas las entradas tienen como registration_type el string “Regular”.

CE_V_08 → Se trata de una clase válida, donde todas las entradas tienen como registration_type el string “Family”.

CE_NV_09 → Se trata de una clase inválida, donde todas las entradas tienen como registration_type es cualquier otro string.

CE_V_10 → Se trata de una clase válida, donde todas las entradas tienen como name_surname un string de 1 a 30 caracteres, en dos o más cadenas separadas por un blanco.

CE_NV_11 → Se trata de una clase inválida, donde todas las entradas tienen como name_surname un string con menos de dos cadenas, sin estar separadas por un blanco.

CE_NV_12 → Se trata de una clase inválida, donde todas las entradas tienen como name_surname un string con más de 30 caracteres.

CE_NV_13 → Se trata de una clase inválida, donde todas las entradas tienen como name_surname un string con menos de 1 caracter.

CE_V_14 → Se trata de una clase válida, donde todas las entradas tienen como phone_number un número entero de 9 dígitos. El rango de valores de entrada comprende todos los números entre el 000000000 y el 999999999.

CE_NV_15 → Se trata de una clase inválida, dónde todas las entradas tienen como phone_number un número entero de menos de 9 dígitos.

CE_NV_16 → Se trata de una clase inválida, dónde todas las entradas tienen como phone_number un número entero de más de 9 dígitos.

CE_NV_17 → Se trata de una clase inválida, donde todas las entradas tienen como phone_number cualquier variable que no sea un número entero.

CE_V_18 → Se trata de una clase válida, donde todas las entradas tienen como age un número entero (int) entre 6 y 125 (siendo este el rango de entrada).

CE_NV_19 → Se trata de una clase inválida, donde todas las entradas tienen como age un número menor que 6.

CE_NV_20 → Se trata de una clase inválida, donde todas las entradas tienen como age un número mayor que 125.

CE_NV_21 → Se trata de una clase inválida, donde todas las entradas tienen como age cualquier variable distinta a un string de números (string con letras).

En cuanto a las salidas, se espera obtener una cadena hexadecimal, por lo que el número de valores de salida también será uno. Además, los datos se irán almacenando en un fichero, por lo que también lo tendremos que comprobar.

CE_V_22 → Se trata de una clase válida, donde todas las salidas son una cadena en formato hexadecimal obtenidas mediante el algoritmo MD5.

CE_NV_23 → Se trata de una clase inválida, donde todas las salidas serían una cadena que no sigue el formato hexadecimal MD5. No es posible crear un test que contemple esta clase de equivalencia, puesto que el método no generará ninguna cadena "incorrecta", simplemente mostrará una excepción si algún dato de la entrada era incorrecto.

CE_V_24 → Se trata de una clase válida, donde todas las salidas son los datos del paciente que se almacenan en el fichero. No definimos un test concreto para esta clase de equivalencia puesto que ya está definido en todos los test válidos anteriores, al comprobarse simultáneamente si los datos correctos introducidos se han almacenado en el fichero.

CE_NV_25 → Se trata de una clase inválida, donde los datos introducidos no son correctos, y por tanto no se almacenan los datos del paciente en el fichero.

CE_NV_26 → Se trata de una clase inválida, donde los datos introducidos son correctos, pero no se almacenan en el fichero. Este caso solo se dará si los datos ya estaban guardados en el fichero.

CE_NV_27 → Se trata de una clase inválida, donde los datos introducidos no son correctos, pero se almacenan en el fichero. Este caso no se va a dar nunca, ya que, si los datos son incorrectos, no llegarán a almacenarse, por lo que no es posible comprobarlo con un test (funcionamiento definido en el propio método).

Análisis de valores límite

patient_id → Dado que debe ser un string de 32 caracteres, los valores límite serán 31, 32 y 33 caracteres.

registration_type → No existen valores límite ya que solo puede ser un string del tipo "Family" o "Regular".

name_surname → Los valores límite serán 0, 1 y 2 blancos, ya que debe tener 1 como mínimo. Por otra parte 0, 1, 2, 29, 30 y 31 caracteres, ya que debe estar entre 1 y 30.

phone_number → Los valores límite serán 11, 12 y 10 caracteres porque ha de ser un número de 9 dígitos, teniendo en cuenta el prefijo + (11 caracteres en total).

age → Los valores límite serán 5, 6, 7, 124, 125 y 126, ya que debe ser un número entre 6 y 125.