

# **EJERCICIO GUIADO 3.1**

## **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 y análisis de valores límite

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, donde todas las entradas tienen como patient\_id un uuid válido (string hexadecimal) 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.

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 30 caracteres o menos, 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\_V\_13 → Se trata de una clase válida, donde todas las entradas tienen como phone\_number un número entero (int) de 9 dígitos. El rango de valores de entrada comprende todos los números entre el 000000000 y el 999999999.

CE\_NV\_14 → Se trata de una clase inválida, donde todas las entradas tienen como phone\_number un número entero (int) de menos de 9 dígitos.

CE\_NV\_15 → Se trata de una clase inválida, donde todas las entradas tienen como phone\_number un número entero (int) de más de 9 dígitos.

CE\_NV\_16 → Se trata de una clase inválida, donde todas las entradas tienen como phone\_number cualquier variable que no sea un número entero.

CE\_V\_17 → 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\_18 → Se trata de una clase inválida, donde todas las entradas tienen como age un número menor que 6.

CE\_NV\_19 → Se trata de una clase inválida, donde todas las entradas tienen como age un número mayor que 125.

CE\_NV\_20 → 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\_21 → Se trata de una clase válida, donde todas las salidas son una cadena en formato hexadecimal obtenidas mediante el algoritmo MD5.

CE\_NV\_22 → 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\_23 → 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\_24 → 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\_25 → 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\_26 → 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).