Corrección cruzada

Desarrollo de Software

ID Grupo: 01
Grupo corregido: 02

Alberto Penas Díaz
NIA: 100471939
Correo:100471939@alumnos.uc3m.es
Grupo: 80
Titulación: Ingeniería informática

Natalia Rodríguez Navarro
NIA: 100471976
Correo: 100471976@alumnos.uc3m.es
Grupo: 80
Titulación: Ingeniería informática

Índice

INTRODUCCIÓN

Este documento recoge la corrección de los casos de prueba definidos por el **grupo 02** sobre el RF1, es decir, el método *register_order()*.

Para realizarla de una forma estructurada, hemos decidido dividirla en tres secciones: Errores conceptuales (a la hora de definir los casos de prueba) , Errores de presentación (del documento) y Preferencias personales (lo que en nuestra opinión sería mejor).

CORRECCIÓN

Errores conceptuales

En primer lugar, el argumento “product_id” es un EAN13, lo que implica que tiene una longitud de 13 y son necesarias las pruebas sobre sus valores límites. En este caso, no han añadido dichas pruebas.

Además, han olvidado añadir el valor límite de “límite inferior” en los tests que comprueban la validez de la dirección.

También, en el apartado que define las pruebas para el “zip_code” (código postal), no han tenido en cuenta que en España, un codigo postal solo es válido si sus dos primeros dígitos están entre los números 01 y 52, ya que éstos hacen referencia al número de provincia en el que se encuentre la dirección (por ejemplo, en el caso de Madrid es 28 y en caso de Barcelona es 08).

Por otro lado, han decidido hacer pruebas funcionales para la salida de la primera función (un código HASH MD5), lo cual no es necesario puesto que este código es implementado por una librería externa ajena a nuestro código. En el supuesto caso de que tuviéramos que haber implementado nosotros la creación de este código, no solamente se tendría que haber comprobado que la longitud del mismo fuera correcta (32 caracteres) si no que también habría que verificar que estuviera en formato hexadecimal, contemplando como válidas las letras de la a **a** la **f** junto con los dígitos del **0** al **9**, no solo la longitud.

Por último, es importante mencionar que en la descripción de los casos nos encontramos con frases poco precisas como “demasiado corto”, lo que no solo dificulta el entendimiento al lector sino que probablemente sea confuso también para los creadores del *excel*.

Errores de presentación

Cabe destacar la dificultad que nos ha generado el hecho de que los distintos valores límites y clases de equivalencia no estuvieran descritos en el lugar correcto del documento, si bien es verdad que la mayoría de estos son correctos y están bien ejecutados, es confuso no saber por qué un caso de prueba es válido si no se provee de una correcta descripción en el lugar adecuado.

Además, considerando errores de presentación aquellos que dificultan (a nuestro parecer) la comprensión lectora del documento, hemos considerado los siguientes errores:

1. Los saltos de línea no son consistentes, a veces son 3 y otras veces son 2.
2. No se define un estilo fijo al escribir determinados tests, unos con mayúsculas y otros con minúsculas.
3. Han borrado algunas líneas de la cuadrícula lo que dificulta la lectura por filas, puesto que son líneas considerablemente largas.

Preferencias personales

Aunque personalmente también nos resulte difícil realizar un proyecto de programación en un mismo idioma (sobre todo cuando el idioma de las definiciones del enunciado es distinto nuestro idioma nativo), consideramos imprescindible aclarar y definir un idioma para el mismo documento, ya que de no hacerlo, esto puede llevar a fallos de comprensión y dificultad en la lectura y entendimiento de los puntos a exponer.

Por último, hubiera estado bien indicar los casos que son abordados por pruebas anteriores, de esta forma podrían evitar de manera bastante eficiente repetir pruebas sin querer en la implementación.