Pruebas de Sistema de Gestión de Cafetería

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Puebla

Erwin Porras Guerra

A01734881

Desarrollo e Implantación de Sistemas de Software

Puebla, Pue, Abril 07, 2024

A01734881@tec.mx

Descripción del Problema:

Se requiere desarrollar un módulo dentro de un sistema de software para cafeterías que permita la incorporación de nuevas bebidas. El enfoque de esta asignación está en la lógica de backend que maneja el ingreso de nuevas bebidas, específicamente la validación del formato de entrada. La interfaz de usuario (UI) no es una preocupación en este punto.

Especificación de Entrada:

- El nombre de la bebida debe contener caracteres alfabéticos y tener una longitud de 2 a 15 caracteres.
- El tamaño de la bebida puede variar, permitiendo un máximo de cinco tamaños por artículo, y debe ser un valor entero dentro del rango de 1 a 48.
- Los tamaños deben ingresarse en orden ascendente (los más pequeños primero).
- El nombre del artículo se debe ingresar primero, seguido de una coma y luego una lista de tamaños.
- Una coma separará cada tamaño.
- Se ignorarán los espacios en blanco en cualquier lugar de la entrada.

Criterios de Aceptación:

- El nombre del artículo es alfabético (válido)
- El nombre del artículo tiene menos de 2 caracteres de longitud (inválido)
- El nombre del artículo tiene de 2 a 15 caracteres de longitud (válido)
- El valor del tamaño está en el rango de 1 a 48 (válido)
- El valor del tamaño es un número entero (válido)
- Los valores del tamaño se ingresan en orden ascendente (válido)
- Se ingresan de uno a cinco valores de tamaño (válido)
- El nombre del artículo es el primero en la entrada (válido)
- Una sola coma separa cada entrada en la lista (válido)
- La entrada contiene o no espacios en blanco (a especificar en las pruebas)

Código desarrollado:

```python
def newbebida(input):
    correctedinput = input.replace(" ","") #clears each " " from the input
    parameters = correctedinput.rsplit(",") # splits on each comma
    if len(parameters) < 2 or len(parameters) > 6: return False # checks for number of inputs, one name and at least
    if any(i.isdigit() for i in parameters[0]): return False # Checks for numbers in parameter 0
    if len(parameters[0]) > 15 or len(parameters[0]) < 2: return False # Checks for name of size between 2 and 15
    try:
        if int(parameters[1]) not in range(1,49): return False #checks for size of first int
        if(len(parameters)) > 2: #if it has more than 1 size
            for i in range (2,len(parameters)):
                if int(parameters[i]) not in range(1,49) or int(parameters[i]) <= int(parameters[i-1]): return False
    except: return False
    return True
```

Documentación de Pruebas:

Por cada criterio de aceptación se realizaron múltiples pruebas, se aseguró que se hicieran pruebas que retornaran un valor positivo que pasara con los criterios, y otras que retornaran un valor negativo que fallara el criterio de aceptación siendo probado:

1.
```python
#El nombre del artículo es alfabético (válido)

def test1(self):
    self.assertEqual(newbebida("aguita,1"), True)


def test2(self):
    self.assertEqual(newbebida("4gu1t4,1"), False)
```

2.
```python
#El nombre del artículo tiene menos de 2 caracteres de longitud (inválido)

def test3(self):
    self.assertEqual(newbebida("normal,1"), True)

def test4(self):
    self.assertEqual(newbebida("a,1"), False)
```

3.
```
#El nombre del artículo tiene de 2 a 15 caracteres de longitud (válido)

def test5(self):
    self.assertEqual(newbebida("Aguita,1"), True)

def test6(self):
    self.assertEqual(newbebida("aguita de limon con poca azucar,1"), False)

def test7(self):
    self.assertEqual(newbebida("abcdefghijklmnop,1"), False)
```

4.
```
#El valor del tamaño está en el rango de 1 a 48 (válido)

def test8(self):
    self.assertEqual(newbebida("aguita,1"), True)

def test9(self):
    self.assertEqual(newbebida("aguita,0"), False)

def test10(self):
    self.assertEqual(newbebida("aguita,48"), True)

def test11(self):
    self.assertEqual(newbebida("aguita,49"), False)
```

5.
```
#El valor del tamaño es un número entero (válido)

def test12(self):
    self.assertEqual(newbebida("aguita,10"), True)

def test13(self):
    self.assertEqual(newbebida("aguita,-1"), False)

def test14(self):
    self.assertEqual(newbebida("aguita,-10"), False)
```

6.
```
#Los valores del tamaño se ingresan en orden ascendente (válido)

def test15(self):
    self.assertEqual(newbebida("aguita,1,2,3"), True)

def test16(self):
    self.assertEqual(newbebida("aguita,3,2,1"), False)

def test17(self):
    self.assertEqual(newbebida("aguita,1,3,2"), False)
```

7.
```
#Se ingresan de uno a cinco valores de tamaño (válido)

def test18(self):
    self.assertEqual(newbebida("aguita,1,2,3,4,5"), True)

def test19(self):
    self.assertEqual(newbebida("aguita"), False)

def test20(self):
    self.assertEqual(newbebida("aguita,1,2,3,4,5,6"), False)
```

8.
```
#El nombre del artículo es el primero en la entrada (válido)

def test21(self):
    self.assertEqual(newbebida("1,aguita"), False)

def test22(self):
    self.assertEqual(newbebida("aguita,1"), True)
```

9.
```
#Una sola coma separa cada entrada en la lista (válido)

def test23(self):
    self.assertEqual(newbebida("aguita,1,2"), True)

def test24(self):
    self.assertEqual(newbebida("aguita,,1"), False)

def test25(self):
    self.assertEqual(newbebida("aguita,1,2,,"), False)
```

10.
```python
#La entrada contiene o no espacios en blanco (a especificar en las pruebas)
# (Ignora los espacios)

def test26(self):
    self.assertEqual(newbebida("a g u i t a , 1 "), True)

def test27(self):
    self.assertEqual(newbebida("aguita, 1, 2, 3, 4, 5"), True)
```

Se hicieron un total de 27 pruebas para probar la funcionalidad completa del programa, el resultado fue el siguiente:

```
PS C:\Users\erwin\Desktop\Semestre6\Tarea1Pruebas> py .\testt1.py
...........................
----------------------------------------------------------------------
Ran 27 tests in 0.001s

OK
```

Todas las pruebas fueron exitosas y se comprobó la funcionalidad perfecta del programa.