Proyecto Final de Investigación: Avance 1

Chavarria Peña Jonathan Andrés
Estudiante Ing. en Sistemas de Computación
Universidad Fidélitas
San José, Costa Rica
jonach1998@gmail.com

Morales Cordero Valeria
Estudiante Ing. en Sistemas de Computación
Universidad Fidélitas
San José, Costa Rica
valemc0603@gmail.com

Phillips Tencio Edmond
Estudiante Ing. en Sistemas de Computación
Universidad Fidélitas
Alajuela, Costa Rica
ephillips10986@ufide.ac

Sánchez Camacho Carlos Daniel
Estudiante Ing. en Sistemas de Computación
Universidad Fidélitas
San José, Costa Rica
csanchez20965@ufide.ac

1. DESARROLLO

2. INVESTIGACIÓN DE LA TECNOLOGÍA

2.1. *Unit Testing*

En el siguiente proyecto se realizarán pruebas a un programa, las mismas se harán utilizando unit testing; pero para poder utilizarlo es importante comprender qué es y cómo utilizarlo. El unit test se define, como el código necesario para comprobar que el código del programa principal esté funcionando como esperábamos. Los unit test son una de muchas pruebas que se pueden realizar para comprobar que los programas estén en funcionamiento. Los unit test se conforman de pequeños tests que comprueban que cada parte de los requisitos del código estén correctos; asimismo, se verifican sus resultados. A la hora de realizar un unit test se puede dividir por partes especificas (Organizar, actuar y afirmar) cada "función" o "caso" que se va a realizar, estas son las siguiente:

- Arrange: Esta primera parte del caso a testear es donde se deben definir las variables o requisitos que necesita el programa para funcionar.
- Act: Esta parte consiste en llamar a los métodos o funciones que se desean probar del código del programa principal a testear.
- Asert: En la última sección se prueba si los resultados son correctos o incorrectos. Dependiendo del resultado, si son correctos se valida y continúa con los otros casos, o se repara, no se continua hasta que el error desaparezca.

Estas partes pueden cambiar de nombre dependiendo de donde se investigue, otros nombres que reciben son Given, When, Then (Dado que, cuando, entonces). Para la última parte del caso (Asert o Then), si hay errores de integración es necesario investigar si se necesitan otros tipos de pruebas de software y de esta manera lograr comprobar la efectividad total del código. Al hacer unit testing se asegura que cada parte el código esta bien y es útil. Es importante saber que los fallos y errores son inevitables, por esto mismo los unit test no se pueden considerar como opcionales. Ya que una aplicación, sitio web, programa o código sin pruebas se puede considerar como inestable, voluble o deficiente. Las pruebas pueden ser desarrolladas por los desarrolladores, mismos que conocen bien el código o también en muchas empresas también las pueden realizar los responsables de QA.

3. SOFTWARE A UTILIZAR

El software a utilizar en la presente investigación es Python -m unittes, es el módulo unittest, este ofrece la posibilidad de crear las pruebas implementando una clase llamada unittest.TestCase en la que se incluirán métodos de pruebas. Tales como los siguientes:

ARREGLAR ESTA TABLA, NO SCREENSHOTS

Método	Comprueba que	Nuevo en
`assertEqual(a, b)`	`a == b`	
`assertNotEqual(a, b)`	`a != b`	
`assertTrue(x)`	`bool(x) is True`	
`assertFalse(x)`	`bool(x) is False`	
`assertIs(a, b)`	`a is b`	3.1
`assertIsNot(a, b)`	`a is not b`	3.1
`assertIsNone(x)`	`x is None`	3.1
`assertIsNotNone(x)`	`x is not None`	3.1
`assertIn(a, b)`	`a in b`	3.1
`assertNotIn(a, b)`	`a not in b`	3.1
`assertIsInstance(a, b)`	`isinstance(a, b)`	3.2
`assertNotIsInstance(a, b)`	`not isinstance(a, b)`	3.2

Figura 1. Métodos de unittest.

El modulo de unit test de python permite utilizar distintos contenedores al realizar pruebas unitarias, como por ejemplo: list, dict y set.

Cada una de las pruebas puede devolver tres respuestas dependiendo del resultado, así como las siguientes:

- OK: Para mostrar que la prueba se ha completado con éxito.
- FAIL: Para mostrar que la prueba no ha pasado exitosamente y se lanza una excepción como esta: AssertionError (sentencia verdadero-falso)
- ERROR: Para dar a entender que la prueba no ha pasado exitosamente, pero el resultado en lugar de ser una aserción es un error.

Unittest.TestCase este incluye la cantidad de tiempo que tomaron las pruebas, junto con un indicador de estado para cada prueba.

3.1. *Escritura de pruebas unitarias para el paquete test*

Se prefiere que las pruebas que utilizan el módulo unittest sigan algunas pautas. Una es nombrar el módulo de prueba comenzando con test y terminarlo con el nombre del módulo que se está probando. Los métodos de prueba en el módulo de prueba deben

comenzar con test y terminar con una descripción de lo que el método está probando. Esto es necesario para que el controlador de prueba reconozca los métodos como métodos de prueba. Por lo tanto, no se debe incluir una cadena de caracteres de documentación para el método. Se debe usar un comentario (como Tests function returns only True or False) para proporcionar documentación para los métodos de prueba. Esto se hace porque las cadenas de documentación se imprimen si existen y, por lo tanto, no se indica qué prueba se está ejecutando.

3.2. Plantilla básica para realizar unit test

```
import unittest
from test import support

class MyTestCase1(unittest.TestCase):

    # Only use setUp() and tearDown() if necessary

    def setUp(self):
        ... code to execute in preparation for tests ...

    def tearDown(self):
        ... code to execute to clean up after tests ...

    def test_feature_one(self):
        # Test feature one.
        ... testing code ...

    def test_feature_two(self):
        # Test feature two.
        ... testing code ...

    ... more test methods ...

class MyTestCase2(unittest.TestCase):
    ... same structure as MyTestCase1 ...

... more test classes ...

if __name__ == '__main__':
    unittest.main()
```

4. Programa a probar

El código al que se le realizarán pruebas será desarrollado en Python, este programa solicita al usuario que ingrese la cédula de la persona que desea buscar y la fecha de nacimiento de la misma, esta información se utilizará para encontrar los datos de la persona en una base de datos ya establecida. Al encontrar la información se imprime en pantalla la siguiente información: saludo, nombre completo, edad, centro de votación y los candidatos oficiales a presidencia y los posibles candidatos. Siendo esta ultima información recolectada desde Wikipedia. La base de datos estará ubicada en el mismo directorio raíz donde esta el programa, si este se borra o se le modifica el nombre, el programa no funcionará.

La idea de este programa es lograr proporcionar de manera fácil información para los votantes. Ya que fácilmente pueden conocer en que región deben votar y los actuales candidatos, además de posibles candidatos a presidencia.

```
C:\Python39\python.exe C:/git/calidad_software_proyecto/Proyecto_Final/Codigo/Votaciones.
    ↪ py
Favor ingrese su cedula sin guiones y con los 0 respectivos: 117110446
Favor ingresar su fecha de nacimiento en el formato dd/mm/yyyy: 13/06/1998
Hola JONATHAN ANDRES
Su nombre completo es: JONATHAN ANDRES CHAVARRIA PENA
Su edad es: 23
Su centro de votacion se ubica en:
       Provincia: SAN JOSE
       Canton: MONTES DE OCA
       Distrito: LOURDES
La lista de candidatos es la siguiente:
                    Partido                  Candidato    Tipo de candidato
0          Liberacion Nacional  Jose Maria Figueres Olsen    Candidato Oficial
1              Nueva Republica    Fabricio Alvarado Munoz    Candidato Oficial
3  Accesibilidad Sin Exclusion   oscar Andres Lopez Arias    Candidato Oficial
4             Accion Ciudadana   Marcia Gonzalez Aguiluz   **Posible Candidato
5             Accion Ciudadana  Carolina Hidalgo Herrera   **Posible Candidato
6             Accion Ciudadana     Welmer Ramos Gonzalez   **Posible Candidato
7             Accion Ciudadana     Hernan Solano Venegas   **Posible Candidato
8             Accion Ciudadana     Martha Zamora Castillo  **Posible Candidato
9       Movimiento Libertario    Carlos Valenciano Kamer    Candidato Oficial

Process finished with exit code 0
```

5. Pruebas a realizar

Para el proyecto necesitamos saber los casos específicos que vamos a probar en nuestro software por lo que definimos los siguientes:

1. Probar el caso en el que todo salga bien
2. Probar si la base de datos ”Distelec.txt”tiene un formato incorrecto
3. Probar si la base de datos ”PADRON_COMPLETO.txt”tiene un formato incorrecto
4. Probar si se ingresa la cédula con letras o caracteres especiales.
5. Probar si no se encuentra una cedula en la base de datos
6. Probar si se deja alguno de los datos solicitados en blanco
7. Probar si no existe el archivo de la base de datos ”Distelec.txt”
8. Probar si no existe el archivo de la base de datos ”PADRON_COMPLETO.txt”
9. Probar si la edad se ingreso en el formato correcto (prueba puede ser: mm/dd/yyyy)
10. Probar si la edad contiene letras o caracteres especiales
11. Probar si la pagina es incorrecta
12. Probar si la pagina no se encuentra

Referencias

[1] Giuseppe Vetri (2020). *Que es un unit test (Prueba unitaria).* https://dev.to/codingpizza/que-es-un-unit-test-prueba-unitaria-2dnk

[2] Yeeply. (2021). *¿Qué son las pruebas unitarias y cómo llevar una a cabo?.* https://www.yeeply.com/blog/que-son-pruebas-unitarias/

[3] Héctor Costa Guzmán (2018). *Unittest.* https://docs.hektorprofe.net/python/documentacion-y-pruebas/unittest/

[4] Python Software Foundation (2018). *Línea de comandos y entorno.* https://docs.python.org/es/3.10/using/cmdline.html#cmdoption-m