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

***Resumen*—Abstract**
***Abstract*—Abstract**
***Keywords*—** one, two, three, four
***Palabras clave*—** uno, dos, tres, cuatro

1. Investigación de la tecnología

A. 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.

2. 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:

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.

A. 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.

B. 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()
```

3. 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
```

4. 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

5. Plan de pruebas

Requerimientos de desarrollo		•
Funcionalidades nuevas		•
Funcionalidades existentes		•
Estrategia de pruebas		•
Pruebas funcionales		
Pruebas no funcionales		
Criterios de inicio	Criterios de suspension	Criterios de aceptación
Entornos y ambientes		•
Necesidades		•
Metodologia		•

6. Casos de prueba

A. Caso 1

- Nombre/Identificador: Todo sale bien.
- Descripción: El usuario digita bien toda la información.
- Objetivo de la prueba: Comprobar el buen funcionamiento del programa.
- Requerimientos o pre condiciones: Estar incluido en la lista de votantes del país.
- Pasos a seguir:
  1) Escribir la cédula, en el espacio correspondiente.
  2) Escribir la fecha de nacimiento, en el espacio correspondiente.
- Resultados esperados: Despliegue de toda la información solicitada.
- Prioridad: Alta.

B. Caso 2

- Nombre/Identificador: ”Distelec.txt” formato incorrecto.
- Descripción: Verificar que el archivo Distelec este separado por otro carácter que no sea comas.
- Objetivo de la prueba: Comprobar que el programa falle cuando el archivo no esta separado por comas.
- Requerimientos o pre condiciones: Archivo en el formato incorrecto.
- Pasos a seguir:
  1) Buscar el archivo ”Distelec.txt” en la carpeta donde esta el programa.
  2) Separar la información del archivo, utilizando como referencia la ”,” para separar.
- Resultados esperados: No se pueda correr el programa.
- Prioridad: Alta

C. Caso 3

- Nombre/Identificador: ”PADRON_COMPLETO.txt” formato incorrecto.
- Descripción: Verificar que el archivo PADRON_COMPLETO este separado por otro carácter que no sea comas.
- Objetivo de la prueba: Comprobar que el programa falle cuando el archivo no esta separado por comas.
- Requerimientos o pre condiciones: Archivo en el formato incorrecto.
- Pasos a seguir:
  1) Buscar el archivo ”PADRON_COMPLETO.txt” en la carpeta donde esta el programa.
  2) Separar la información del archivo, utilizando como referencia la ”,” para separar.

- Resultados esperados: No se pueda correr el programa.
- Prioridad: Alta

D. Caso 4

- Nombre/Identificador: Cédula con letras o caracteres especiales.
- Descripción: El usuario digita la cédula utilizando letras o caracteres especiales.
- Objetivo de la prueba: No poder correr el programa.
- Requerimientos o pre condiciones: No hay.
- Pasos a seguir:
  1) Escribir la cédula conteniendo letras o caracteres especiales, en el espacio correspondiente.
- Resultados esperados: No corra el programa.
- Prioridad: Alta

E. Caso 5

- Nombre: Cédula no encontrada.
- Descripción: Probar si no se encuentra una cédula en la base de datos.
- Objetivo de la prueba: Identificar que sucede si no se encuentra una cédula ingresada.
- Requerimientos: Ingresar una cédula, base de datos.
- Pasos para seguir:
  1) Ingresar una cédula en el campo correspondiente y ver el resultado.
- Resultados esperados: Al ingresar la cédula no existente en la base de datos dar un mensaje de advertencia y no dejar continuar.
- Prioridad: Alta.

F. Caso 6

- Nombre: Fecha nacimiento vacío.
- Descripción: Probar si se deja alguno de los datos solicitados en blanco.
- Objetivo de la prueba: Ver el resultado si se deja uno de los datos solicitados en blanco.
- Requerimientos: Base de datos.
- Pasos a seguir:
  1) Ingresar los datos y probar dejando algunos espacios en blanco para ver el resultado
- Resultados esperados: Al dejar uno o varios espacios en blanco el sistema da un mensaje de advertencia y no lo se puede continuar.
- Prioridad: Alta.

G. Caso 7

- Nombre: ”Distelec.txt” no existe.
- Descripción: Probar si no existe el archivo de la base de datos ”Distelec.txt”.
- Objetivo de la prueba: Ver si no existe el archivo ”Distelec.txt” de la base de datos que sucede con el programa.
- Requerimientos: Documento ”Distelec.txt”.
- Pasos a seguir:
  1) Verificar la existencia o no del archivo ”Distelec.txt” en la base de datos y ejecutar el programa.
- Resultados esperados: Al no encontrar el archivo de base de datos ”Distelec.txt” no dejar continuar e indicarlo con un mensaje para continuar con el programa.
- Prioridad: Muy alta.

H. Caso 8

- Nombre: ”PADRON_COMPLETO.txt” no existe.
- Descripción: Probar si no existe el archivo de la base de datos ”PADRON_COMPLETO.txt”.
- Objetivo de la prueba: Verificar lo que sucede si no se encuentra en el programa el archivo de la base de datos ”PADRON_COMPLETO.txt”.
- Requerimientos: Documento ”PADRON_COMPLETO.txt”.
- Pasos a seguir:
  1) Ejecutar el programa y ver que sucede si no se encuentra el archivo ”PADRON_COMPLETO.txt”.
- Resultados esperados: Al no encontrar el archivo ”PADRON_COMPLETO.txt” no dejar continuar con el programa e indicar el error con un mensaje.
- Prioridad: Muy alta.

I. Caso 9

- Nombre: Formato fecha nacimiento incorrecto.
- Descripción: Probar si la edad se ingresó en el formato incorrecto.
- Objetivo de la prueba: verificar que la edad no se ingreso en el formato deseado.
- Requerimientos: Ingresar fecha de nacimiento, base de datos.
- Pasos por seguir:
  1) Ingresar la fecha de nacimiento en formato incorrecto.
- Resultados esperados:
- Prioridad: Alta.

J. Caso 10

- Nombre: Fecha nacimiento caracteres especiales no esperados.
- Descripción:
- Probar si la edad contiene letras o caracteres especiales.
- Objetivo de la prueba: Ver que sucede si se ingresa la edad con caracteres no esperados por el programa (letras o caracteres especiales) datos.
- Requerimientos: Ingresar la edad.
- Pasos por seguir:
  1) Ingresar la edad en el sistema con letras o caracteres especiales y comprobar que sucede.
- Resultados esperados: Al ingresar letras o caracteres incorrectos el sistema debe de mostrar un mensaje de error que ingrese la edad solo con números.
- Prioridad: Media

K. Caso 11

- Nombre: Página incorrecta
- Descripción: Probar si la página es incorrecta
- Objetivo de la prueba: Ver que sucede al ingresar a una página incorrecta.
- Requerimientos: Página incorrecta.
- Pasos por seguir: Confirmar que sucede si se ingresa a una pagina incorrecta.
- Resultados esperados: Al no encontrar la página correcta debe de mostrar un mensaje de error diciendo que esta página es incorrecta.
- Prioridad: Muy alta.

L. Caso 12

- Nombre: Pagina no existe.
- Descripción: Probar si la página no se encuentra.

- Objetivo de la prueba: Verificar lo que sucede si no se encuentra la pagina.
- Requerimientos: Dirección de página correcta.
- Pasos por seguir:
  1) Ejecutar el programa y ver que sucede si no encuentra la dirección de la página correcta.
- Resultados esperados: Al no encontrar la página debería de dar un mensaje de error y no dejar continuar el programa.
- Prioridad: Muy alta.

7. Pruebas Unitarias

```python
class TestVotaciones(unittest.TestCase):
    def setUp(self) -> None:
        self._target = Votaciones("dummy", "dummy", "dummy")
        self._distelec = """101001,SAN JOSE,CENTRAL,HOSPITAL
        108001,SAN JOSE,CENTRAL,HATILLO"""
        self._padron = """100339724,109007, ,20231119,00000,JOSE,DELGADO,CORRALES
        100842598,108001, ,20261024,00000,CARMEN                    ,CORRALES                    ,MORALES
        101019387,101026, ,20230416,00000,CLAUDIA MANUELA           ,ESPINOZA                    ,FONSECA"""
        self._distelec_return = mock.mock_open(read_data=self._distelec).return_value
        self._padron_return = mock.mock_open(read_data=self._padron).return_value
        self._url_return = url

    # Caso 1
    @mock.patch("Proyecto_Final.Codigo.Votaciones.requests")
    @mock.patch("Proyecto_Final.Codigo.Votaciones.input")
    @mock.patch("Proyecto_Final.Codigo.Votaciones.open")
    @mock.patch('sys.stdout', new_callable=io.StringIO)
    def test_todo_sale_bien(self, mock_stdout, mock_open, mock_input, mock_requests):
        mock_open.side_effect = [self._distelec_return, self._padron_return]
        mock_input.side_effect = ["100842598", "13/12/1998"]
        mock_requests.get.return_value.content = url
        self._target.run()
        target_output = mock_stdout.getvalue()
        expected_output = """Hola CARMEN
Su nombre completo es: CARMEN CORRALES MORALES
Su edad es: 22
Su centro de votacion se ubica en:
        Provincia: SAN JOSE
        Canton: CENTRAL
        Distrito: HATILLO
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
2   Progreso Social Democratico        Rodrigo Chaves Robles   Candidato Oficial
3       Unidad Social Cristiana      Lineth Saborio Chaverri   Candidato Oficial
4                Unidos Podemos         Natalia Diaz Quintana   Candidato Oficial
6     Accesibilidad Sin Exclusion      oscar Andres Lopez Arias   Candidato Oficial
7              Accion Ciudadana      Marcia Gonzalez Aguiluz  **Posible Candidato
8              Accion Ciudadana      Carolina Hidalgo Herrera  **Posible Candidato
9              Accion Ciudadana         Welmer Ramos Gonzalez  **Posible Candidato
10             Accion Ciudadana        Hernan Solano Venegas  **Posible Candidato
11        Movimiento Libertario       Carlos Valenciano Kamer   Candidato Oficial
12             Nueva Generacion              Sergio Mena Diaz   Candidato Oficial
13                Union Liberal      Federico Malavassi Calvo   Candidato Oficial
14                  Por definir   Juan Diego Castro Fernandez  **Posible Candidato
15                  Por definir               Viviam Quesada  **Posible Candidato
17       Coalicion para el Cambio        Eliecer Feinzaig Mintz   Candidato Oficial
18                Frente Amplio  Jose Maria Villalta Florez-Estrada   Candidato Oficial
19         Restauracion Nacional         Eduardo Cruickshank  **Posible Candidato
20         Restauracion Nacional                 Melvin Nunez  **Posible Candidato
"""
        self.assertEqual(expected_output, target_output)

    # Caso 2
    @mock.patch("Proyecto_Final.Codigo.Votaciones.input")
    @mock.patch("Proyecto_Final.Codigo.Votaciones.open")
    def test_distelect_formato_incorrecto(self, mock_open, mock_input):
        distelec = """101001-SAN JOSE-CENTRAL-HOSPITAL
108001-SAN JOSE-CENTRAL-HATILLO"""
        distelec_return = mock.mock_open(read_data=distelec).return_value
        mock_open.side_effect = [distelec_return, self._padron_return]
        mock_input.side_effect = ["100842598", "13/12/1998"]
        with self.assertRaises(KeyError):
            self._target.run()

    # Caso 3
    @mock.patch("Proyecto_Final.Codigo.Votaciones.input")
    @mock.patch("Proyecto_Final.Codigo.Votaciones.open")
    def test_padron_formato_incorrecto(self, mock_open, mock_input):
        padron = """100339724-109007- -20231119-00000-JOSE-DELGADO-CORRALES
100842598-108001- -20261024-00000-CARMEN                    -CORRALES                    -MORALES
101019387-101026- -20230416-00000-CLAUDIA MANUELA           -ESPINOZA                    -FONSECA"""
        padron_return = mock.mock_open(read_data=padron).return_value
        mock_open.side_effect = [self._distelec_return, padron_return]
        mock_input.side_effect = ["100842598", "13/12/1998"]
        with self.assertRaises(ValueError):
            self._target.run()

    # Caso 4
    @mock.patch("Proyecto_Final.Codigo.Votaciones.input")
    @mock.patch("Proyecto_Final.Codigo.Votaciones.open")
    def test_cedula_caracteres_especiales_y_letras(self, mock_open, mock_input):
        mock_open.side_effect = [self._distelec_return, self._padron_return]
        mock_input.side_effect = ["holi151.<?", "13/12/1998"]
        with self.assertRaises(KeyError):
            self._target.run()

    # Caso 5
    @mock.patch("Proyecto_Final.Codigo.Votaciones.input")
    @mock.patch("Proyecto_Final.Codigo.Votaciones.open")
    def test_cedula_no_encontrada(self, mock_open, mock_input):
        mock_open.side_effect = [self._distelec_return, self._padron_return]
        mock_input.side_effect = ["117110446", "13/06/1998"]
        with self.assertRaises(KeyError):
            self._target.run()

    # Caso 6
    @mock.patch("Proyecto_Final.Codigo.Votaciones.input")
    @mock.patch("Proyecto_Final.Codigo.Votaciones.open")
    def test_fecha_nacimiento_vacio(self, mock_open, mock_input):
        mock_open.side_effect = [self._distelec_return, self._padron_return]
        mock_input.side_effect = ["100842598", ""]
        with self.assertRaises(ValueError):
            self._target.run()

    # Caso 7
    def test_distelect_no_existe(self):
        with self.assertRaises(FileNotFoundError):
            self._target.run()

    # Caso 8
    def test_padron_no_existe(self):
        target = Votaciones("Database/Distelec.txt", "dummy", "dummy")
        with self.assertRaises(FileNotFoundError):
            target.run()

    # Caso 9
    @mock.patch("Proyecto_Final.Codigo.Votaciones.requests")
    @mock.patch("Proyecto_Final.Codigo.Votaciones.input")
    @mock.patch("Proyecto_Final.Codigo.Votaciones.open")
    @mock.patch('sys.stdout', new_callable=io.StringIO)
    def test_formato_fecha_nacimiento_incorrecto(self, mock_stdout, mock_open, mock_input, mock_requests):
        mock_open.side_effect = [self._distelec_return, self._padron_return]
        mock_input.side_effect = ["100842598", "13/12/98"]
        mock_requests.get.return_value.content = url
        self._target.run()
        target_output = mock_stdout.getvalue()
        expected_output = """Hola CARMEN
Su nombre completo es: CARMEN CORRALES MORALES
Su edad es: 1922
Su centro de votacion se ubica en:
\tProvincia: SAN JOSE
\tCanton: CENTRAL
\tDistrito: HATILLO
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
2   Progreso Social Democratico        Rodrigo Chaves Robles   Candidato Oficial
3       Unidad Social Cristiana      Lineth Saborio Chaverri   Candidato Oficial
4                Unidos Podemos         Natalia Diaz Quintana   Candidato Oficial
6     Accesibilidad Sin Exclusion      oscar Andres Lopez Arias   Candidato Oficial
7              Accion Ciudadana      Marcia Gonzalez Aguiluz  **Posible Candidato
8              Accion Ciudadana      Carolina Hidalgo Herrera  **Posible Candidato
9              Accion Ciudadana         Welmer Ramos Gonzalez  **Posible Candidato
10             Accion Ciudadana        Hernan Solano Venegas  **Posible Candidato
11        Movimiento Libertario       Carlos Valenciano Kamer   Candidato Oficial
12             Nueva Generacion              Sergio Mena Diaz   Candidato Oficial
13                Union Liberal      Federico Malavassi Calvo   Candidato Oficial
14                  Por definir   Juan Diego Castro Fernandez  **Posible Candidato
15                  Por definir               Viviam Quesada  **Posible Candidato
17       Coalicion para el Cambio        Eliecer Feinzaig Mintz   Candidato Oficial
18                Frente Amplio  Jose Maria Villalta Florez-Estrada   Candidato Oficial
19         Restauracion Nacional         Eduardo Cruickshank  **Posible Candidato
20         Restauracion Nacional                 Melvin Nunez  **Posible Candidato
"""
        self.assertEqual(expected_output, target_output)

    # Caso 10
    @mock.patch("Proyecto_Final.Codigo.Votaciones.input")
    @mock.patch("Proyecto_Final.Codigo.Votaciones.open")
    def test_fecha_nacimiento_caracteres_especiales_no_esperados(self, mock_open, mock_input):
        mock_open.side_effect = [self._distelec_return, self._padron_return]
        mock_input.side_effect = ["100842598", "13-12-1998"]
        with self.assertRaises(ValueError):
            self._target.run()

    # Caso 11
    @mock.patch("Proyecto_Final.Codigo.Votaciones.input")
    @mock.patch("Proyecto_Final.Codigo.Votaciones.open")
    def test_pagina_incorrecta(self, mock_open, mock_input):
        target = Votaciones("dummy", "dummy", "https://es.wikipedia.org/wiki/Elecciones_generales_de_Costa_Rica_de_3022")
        mock_open.side_effect = [self._distelec_return, self._padron_return]
        mock_input.side_effect = ["100842598", "13/06/1998"]
        with self.assertRaises(IndexError):
            target.run()

    # Caso 12
    @mock.patch("Proyecto_Final.Codigo.Votaciones.input")
    @mock.patch("Proyecto_Final.Codigo.Votaciones.open")
    def test_pagina_no_existe(self, mock_open, mock_input):
        mock_open.side_effect = [self._distelec_return, self._padron_return]
        mock_input.side_effect = ["100842598", "13/06/1998"]
        with self.assertRaises(requests.exceptions.MissingSchema):
            self._target.run()
```

References

[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