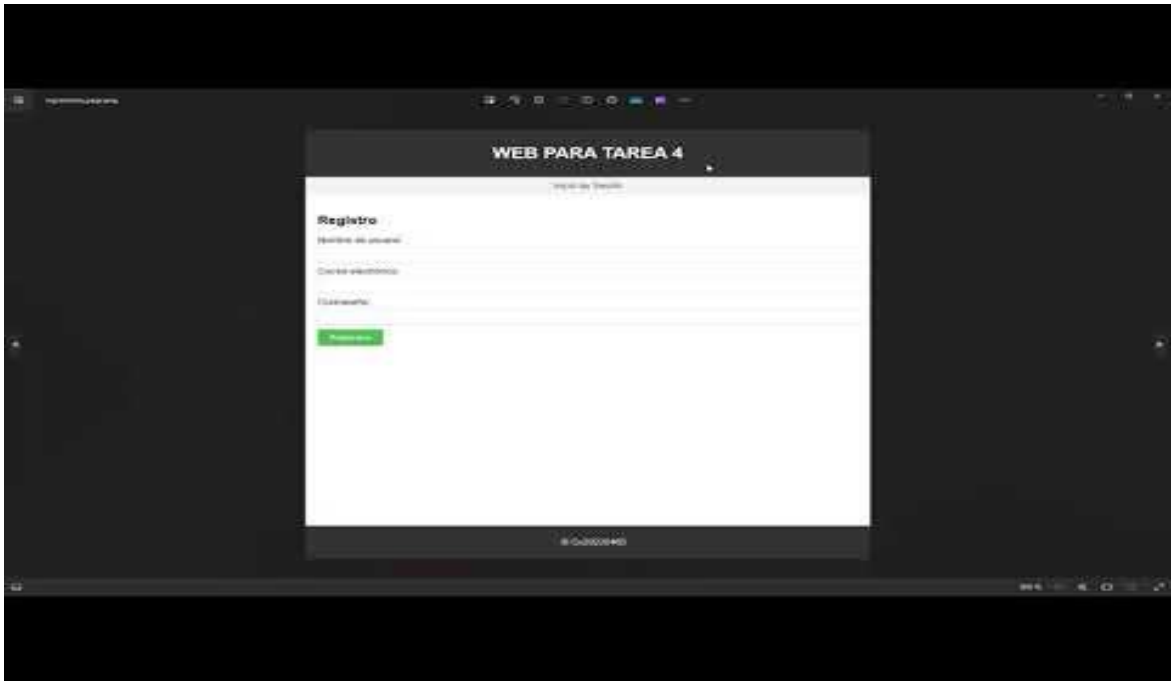
ENLACE DE GIT: https://github.com/SAGITARIO13/WEB_PARA_TAREA-4_CS20220465

ENLACE DE JIRA: https://carlossanchezcoplin.atlassian.net/jira/software/projects/KAN/boards/1

ENLACE DE VIDEO: https://youtu.be/Z4D63wHrRIk



Documentación de la tarea 4:

Prueba_buscar_publicacion

CODIGO

Prueba_buscar_publicacion.py > ...

```python
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import HtmlTestRunner  # Importar HtmlTestRunner para generar informes HTML
import os

class TestSearchPublications(unittest.TestCase):

    def setUp(self):
        # Configuración inicial antes de cada prueba
        self.driver = webdriver.Chrome()  # Iniciar el navegador Chrome
        self.driver.get("http://localhost/TAREA%204/home.php")  # Abrir la URL de la página web

    def test_search_publications(self):
        # Prueba de búsqueda de publicaciones
        self.take_screenshot("search_publications_page")  # Tomar captura de pantalla de la página inicial

        # Encontrar el campo de búsqueda por su ID y escribir un término de búsqueda
        search_input = self.driver.find_element(By.ID, "search")
        search_input.send_keys("Término de búsqueda")

        # Encontrar el botón de búsqueda por su selector CSS y hacer clic
        search_button = self.driver.find_element(By.CSS_SELECTOR, "input[type='submit']")
        search_button.click()

        self.take_screenshot("search_results")  # Tomar captura de pantalla de los resultados de la búsqueda

    def tearDown(self):
        # Limpiar después de cada prueba
        self.driver.quit()  # Cerrar el navegador después de la prueba

    def take_screenshot(self, step_name):
        # Método para tomar una captura de pantalla
        screenshot_folder = "screenshots"  # Carpeta para almacenar capturas de pantalla
        if not os.path.exists(screenshot_folder):
            os.makedirs(screenshot_folder)  # Crear la carpeta si no existe
        filename = f"{screenshot_folder}/{step_name}.png"  # Nombre de archivo para la captura de pantalla
        self.driver.save_screenshot(filename)  # Guardar la captura de pantalla con el nombre dado

if __name__ == "__main__":
    # Ejecutar las pruebas y generar un informe HTML utilizando HtmlTestRunner
    unittest.main(testRunner=HtmlTestRunner.HTMLTestRunner(output='reportes'))

```

Resultados de la prueba unitaria

Resultados De La Prueba Unitaria

Hora de inicio: 2024-04-06 22:25:08

Duración: 3,70 s

Resumen: Total: 1, Aprobado: 1

__main__.TestSearchPublications	Estado
publicaciones_de_búsqueda_prueba	Aprobar

Total: 1, Pase: 1 -- Duración: 3,70 s

Pagina inicial

Crear Publicación Buscar Publicaciones

Publicaciones

Buscar:

Buscar

Título	Contenido	
Nuevo título de la publicación	Nuevo contenido de la publicación	Editar
Título de prueba	Contenido de prueba	Editar
da	sd	Editar
Título de prueba	Contenido de pruebas	Editar
Título de prueba	Contenido de prueba	Editar
Título de prueba	Contenido de prueba	Editar
Título de prueba	Contenido de prueba	Editar
Término de búsqueda	Contenido de prueba	Editar

Resultados de búsqueda

Crear Publicación Buscar Publicaciones

Publicaciones

Buscar:

Buscar

Título	Contenido	
Nuevo título de la publicación	Nuevo contenido de la publicación	Editar
Título de prueba	Contenido de prueba	Editar
da	sd	Editar
Título de prueba	Contenido de pruebas	Editar
Título de prueba	Contenido de prueba	Editar
Título de prueba	Contenido de prueba	Editar
Título de prueba	Contenido de prueba	Editar
Término de búsqueda	Contenido de prueba	Editar

Prueba_editar

CODIGO

Prueba_editar_p.py > ...

```python
import unittest
import os
from selenium import webdriver
from selenium.webdriver.common.by import By
~/Downloads/Nueva carpeta (2)/                tRunner para generar informes HTML
prueba_publicacion.py • 5 problems in this file
class TestEditPublication(unittest.TestCase):

    def setUp(self):
        # Configuración inicial antes de cada prueba
        self.driver = webdriver.Chrome()  # Iniciar el navegador Chrome
        self.driver.get("http://localhost/TAREA%204/home.php")  # Abrir la URL de la página web

    def test_edit_publication(self):
        # Prueba de edición de una publicación
        self.take_screenshot("1_home_page")  # Tomar captura de pantalla de la página inicial

        # Encontrar el enlace de "Editar" por su texto y hacer clic
        edit_link = self.driver.find_element(By.LINK_TEXT, "Editar")
        edit_link.click()

        self.take_screenshot("2_edit_publication_page")  # Tomar captura de pantalla de la página de edición

        # Encontrar el campo de título por su nombre, borrar el contenido existente y enviar un nuevo título
        title_input = self.driver.find_element(By.NAME, "titulo")
        title_input.clear()
        title_input.send_keys("Nuevo título de la publicación")

        # Encontrar el campo de contenido por su nombre, borrar el contenido existente y enviar nuevo conten:
        content_input = self.driver.find_element(By.NAME, "contenido")
        content_input.clear()
        content_input.send_keys("Nuevo contenido de la publicación")

        # Encontrar el botón de "Guardar" por su selector CSS y hacer clic
        save_button = self.driver.find_element(By.CSS_SELECTOR, "input[type='submit']")
        save_button.click()

        self.take_screenshot("3_publication_edited")  # Tomar captura de pantalla después de editar

    def tearDown(self):
        # Limpiar después de cada prueba
        self.driver.quit()  # Cerrar el navegador después de la prueba

    def take_screenshot(self, step_name):
        # Método para tomar una captura de pantalla
        screenshot_folder = "screenshots"  # Carpeta para almacenar capturas de pantalla
        if not os.path.exists(screenshot_folder):
            os.makedirs(screenshot_folder)  # Crear la carpeta si no existe
        filename = f"{screenshot_folder}/{step_name}.png"  # Nombre de archivo para la captura de pantalla
        self.driver.save_screenshot(filename)  # Guardar la captura de pantalla con el nombre dado

if __name__ == "__main__":
    # Ejecutar las pruebas y generar un informe HTML utilizando HtmlTestRunner
    unittest.main(testRunner=HtmlTestRunner.HTMLTestRunner(output='reportes'))
```

1_home_page

Crear Publicación Buscar Publicaciones

Publicaciones

Buscar:

Buscar

Título	Contenido	
Nuevo título de la publicación	Nuevo contenido de la publicación	Editar
Título de prueba	Contenido de prueba	Editar
da	sd	Editar
Título de prueba	Contenido de pruebas	Editar
Título de prueba	Contenido de prueba	Editar
Título de prueba	Contenido de prueba	Editar
Título de prueba	Contenido de prueba	Editar
Término de búsqueda	Contenido de prueba	Editar

2_edit_publication_page

Crear Publicación Buscar Publicaciones

Editar Publicación

Título:

Nuevo título de la publicación

Contenido:

Nuevo contenido de la publicación

Guardar Cambios

3_publication_edited

Crear Publicación Buscar Publicaciones

Publicaciones

Buscar:

Buscar

Titulo	Contenido	
Nuevo título de la publicación	Nuevo contenido de la publicación	Editar
Título de prueba	Contenido de prueba	Editar
da	sd	Editar
Título de prueba	Contenido de pruebas	Editar
Título de prueba	Contenido de prueba	Editar
Título de prueba	Contenido de prueba	Editar
Título de prueba	Contenido de prueba	Editar
Término de búsqueda	Contenido de prueba	Editar

Resultados de la prueba unitaria

Resultados De La Prueba Unitaria

Hora de inicio: 2024-04-06 22:25:55

Duración: 4,10 s

Resumen: Total: 1, Aprobado: 1

__main__.TestEditPublicación	Estado
prueba_edit_publicación	Aprobar

Total: 1, Pase: 1 -- Duración: 4,10 s

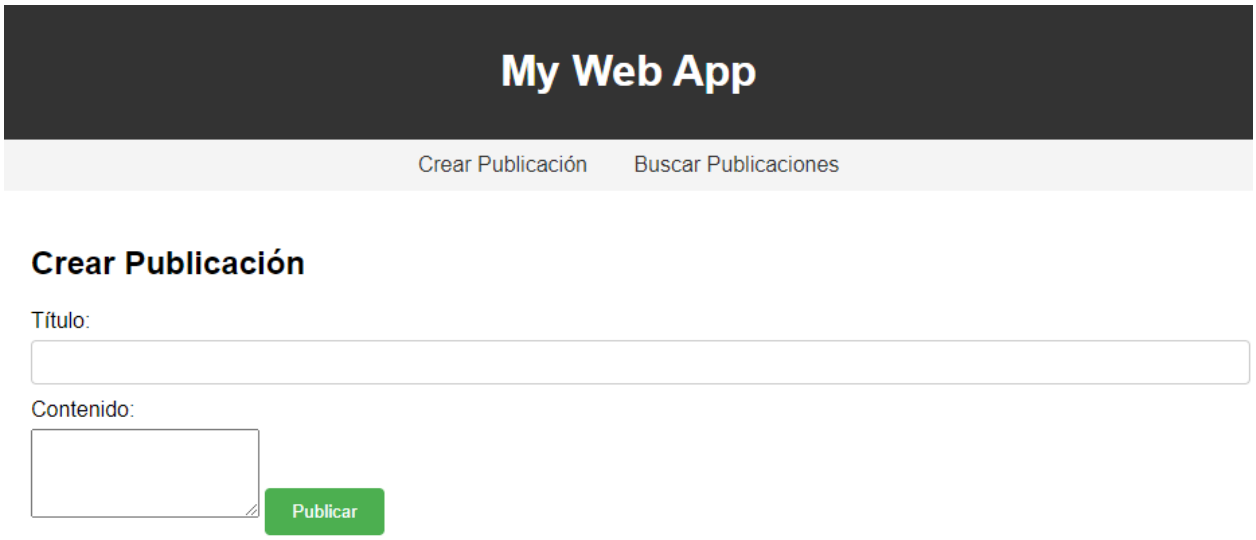
prueba_publicacion

CODIGO

prueba_publicacion.py > ...

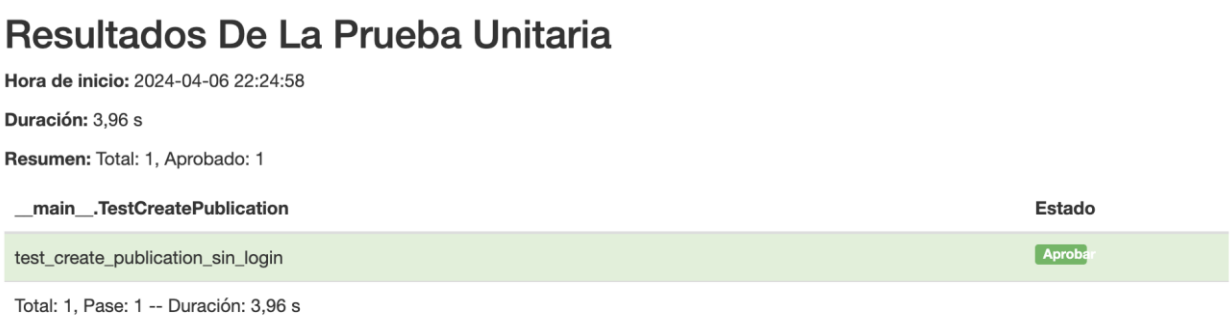
```python
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import HtmlTestRunner  # Importar HtmlTestRunner para generar informes HTML
import os

class TestCreatePublication(unittest.TestCase):

    def setUp(self):
        # Configuración inicial antes de cada prueba
        self.driver = webdriver.Chrome()  # Iniciar el navegador Chrome
        self.driver.get("http://localhost/TAREA%204/crear_publicacion.html")  # Abrir la URL de la página de creación de publicación

    def test_create_publication_without_login(self):
        # Prueba de creación de publicación sin iniciar sesión
        self.take_screenshot("create_publication_page")  # Tomar captura de pantalla de la página de creación de publicación

        # Encontrar los campos de título, contenido y el botón de publicar
        title_input = self.driver.find_element(By.ID, "titulo")
        content_input = self.driver.find_element(By.ID, "contenido")
        publish_button = self.driver.find_element(By.CSS_SELECTOR, "input[type='submit']")

        # Llenar los campos de título y contenido
        title_input.send_keys("Término de búsqueda")
        content_input.send_keys("Contenido de prueba")

        self.take_screenshot("filled_publication_form")  # Tomar captura de pantalla después de llenar el formulario de publicación

        publish_button.click()  # Hacer clic en el botón de publicar

        # Esperar a que la URL sea "http://localhost/TAREA%204/home.php" (redirección después de publicar)
        WebDriverWait(self.driver, 10).until(EC.url_to_be("http://localhost/TAREA%204/home.php"))
        self.take_screenshot("redirected_to_home_page")  # Tomar captura de pantalla después de redirigir a la página de inicio

    def tearDown(self):
        # Limpiar después de cada prueba
        self.driver.quit()  # Cerrar el navegador después de la prueba

    def take_screenshot(self, step_name):
        # Método para tomar una captura de pantalla
        screenshot_folder = "screenshots"  # Carpeta para almacenar capturas de pantalla
        if not os.path.exists(screenshot_folder):
            os.makedirs(screenshot_folder)  # Crear la carpeta si no existe
        filename = f"{screenshot_folder}/{step_name}.png"  # Nombre de archivo para la captura de pantalla
        self.driver.save_screenshot(filename)  # Guardar la captura de pantalla con el nombre dado

if __name__ == "__main__":
    # Ejecutar las pruebas y generar un informe HTML utilizando HtmlTestRunner
    unittest.main(testRunner=HtmlTestRunner.HTMLTestRunner(output='reportes'))

```

create_publication_page



filled_login_form

Resultado de prueba unitaria



prueba_registro

CODIGO

prueba_registro.py > ...

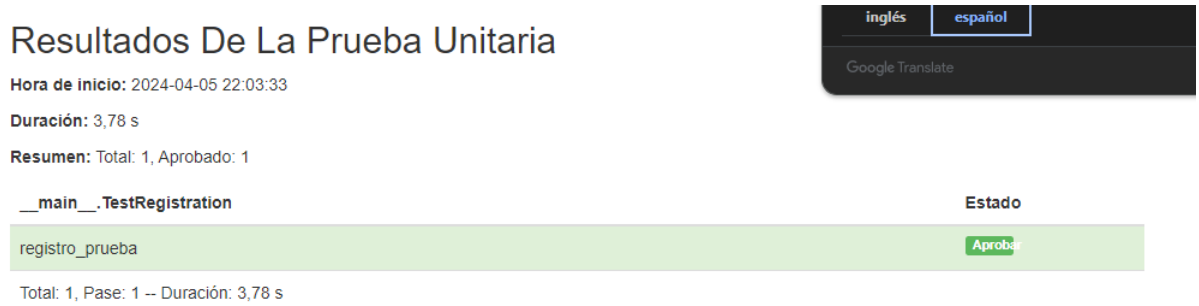
```python
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import HtmlTestRunner  # Importar HtmlTestRunner para generar informes HTML
import os

class TestRegistration(unittest.TestCase):

    def setUp(self):
        # Configuración inicial antes de cada prueba
        self.driver = webdriver.Chrome()  # Iniciar el navegador Chrome
        self.driver.get("http://localhost/TAREA%204/index.html")  # Abrir la URL de la página de registro

    def test_registration(self):
        # Prueba de registro de usuario
        self.take_screenshot("registration_page")  # Tomar captura de pantalla de la página de registro

        # Encontrar los campos de nombre de usuario, correo electrónico, contraseña y el botón de registro
        username_input = self.driver.find_element(By.ID, "username")
        email_input = self.driver.find_element(By.ID, "email")
        password_input = self.driver.find_element(By.ID, "password")
        register_button = self.driver.find_element(By.XPATH, "//input[@value='Registrarse']")

        # Llenar los campos de nombre de usuario, correo electrónico y contraseña
        username_input.send_keys("nombre_de_usuario")
        email_input.send_keys("correo_electronico@example.com")
        password_input.send_keys("contraseña123")

        self.take_screenshot("filled_registration_form")  # Tomar captura de pantalla después de llenar el formulario de registro

        register_button.click()  # Hacer clic en el botón de registro

        # Esperar a que la URL sea "http://localhost/TAREA%204/registro_exitoso.php" (página de registro exitoso)
        WebDriverWait(self.driver, 10).until(EC.url_to_be("http://localhost/TAREA%204/registro_exitoso.php"))
        self.take_screenshot("registration_successful")  # Tomar captura de pantalla después de un registro exitoso

    def tearDown(self):
        # Limpiar después de cada prueba
        self.driver.quit()  # Cerrar el navegador después de la prueba

    def take_screenshot(self, step_name):
        # Método para tomar una captura de pantalla
        screenshot_folder = "screenshots"  # Carpeta para almacenar capturas de pantalla
        if not os.path.exists(screenshot_folder):
            os.makedirs(screenshot_folder)  # Crear la carpeta si no existe
        filename = f"{screenshot_folder}/{step_name}.png"  # Nombre de archivo para la captura de pantalla
        self.driver.save_screenshot(filename)  # Guardar la captura de pantalla con el nombre dado

if __name__ == "__main__":
    # Ejecutar las pruebas y generar un informe HTML utilizando HtmlTestRunner
    unittest.main(testRunner=HtmlTestRunner.HTMLTestRunner(output='reportes'))
```

registration_page

filled_registration_form

registration_successful

Resultado de prueba unitaria



Resultados De La Prueba Unitaria

Hora de inicio: 2024-04-05 22:03:33

Duración: 3,78 s

Resumen: Total: 1, Aprobado: 1

__main__.TestRegistration	Estado
registro_prueba	Aprobar

Total: 1, Pase: 1 -- Duración: 3,78 s

Prueba login

CODIGO

prueba.py > TestLogin > test_successful_login

```python
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import HtmlTestRunner  # Importar HtmlTestRunner para generar informes HTML
import os

class TestLogin(unittest.TestCase):

    def setUp(self):
        # Configuración inicial antes de cada prueba
        self.driver = webdriver.Chrome()  # Iniciar el navegador Chrome
        self.driver.get("http://localhost/Tarea%204/inicio_sesion.html")

    def test_successful_login(self):
        # Prueba de inicio de sesión exitoso
        self.take_screenshot("login_page")

        # Esperar hasta que el campo de nombre de usuario esté presente
        username_input = WebDriverWait(self.driver, 10).until(
            EC.presence_of_element_located((By.ID, "login_email"))
        )
        password_input = self.driver.find_element(By.ID, "login_password")
        login_button = self.driver.find_element(By.CSS_SELECTOR, "input[type='submit']")

        # Llenar los campos de nombre de usuario y contraseña
        username_input.send_keys("carlossanchezcoplin@gmail.com")
        password_input.send_keys("123")
        self.take_screenshot("filled_login_form")  # Tomar captura de pantalla después de llenar el formulario de inicio de sesión
        login_button.click()

        self.take_screenshot("successful_login")  # Tomar captura de pantalla después de un inicio de sesión exitoso

    def test_failed_login(self):
        # Prueba de inicio de sesión fallido
        self.take_screenshot("login_page")  # Tomar captura de pantalla de la página de inicio de sesión

        # Encontrar los campos de nombre de usuario, contraseña y el botón de inicio de sesión
        username_input = self.driver.find_element(By.ID, "login_email")
        password_input = self.driver.find_element(By.ID, "login_password")
        login_button = self.driver.find_element(By.CSS_SELECTOR, "input[type='submit']")

        # Llenar los campos de nombre de usuario y contraseña con credenciales incorrectas
        username_input.send_keys("usuarioincorrecto@example.com")
        password_input.send_keys("contraseñaincorrecta")
        self.take_screenshot("filled_login_form_with_wrong_credentials")
        login_button.click()  # Hacer clic en el botón de inicio de sesión

        # Esperar a que aparezca el mensaje de error de credenciales incorrectas y tomar captura de pantalla
        error_message_element = WebDriverWait(self.driver, 10).until(
            EC.visibility_of_element_located((By.XPATH, "//p[contains(text(), 'Credenciales incorrectas')]"))
        )
        self.take_screenshot("failed_login")  # Tomar captura de pantalla después de un inicio de sesión fallido
        self.assertTrue(error_message_element.is_displayed())  # Verificar que el mensaje de error esté visible en la página

```

```
57      def tearDown(self):
58          # Limpiar después de cada prueba
59          self.driver.quit()  # Cerrar el navegador después de la prueba
60
61      def take_screenshot(self, step_name):
62          # Método para tomar una captura de pantalla
63          screenshot_folder = "screenshots"  # Carpeta para almacenar capturas de pantalla
64          if not os.path.exists(screenshot_folder):
65              os.makedirs(screenshot_folder)  # Crear la carpeta si no existe
66          filename = f"{screenshot_folder}/{step_name}.png"  # Nombre de archivo para la captura de pantalla
67          self.driver.save_screenshot(filename)  # Guardar la captura de pantalla con el nombre dado
68
69  if __name__ == "__main__":
70      # Ejecutar las pruebas y generar un informe HTML utilizando HtmlTestRunner
71      unittest.main(testRunner=HtmlTestRunner.HTMLTestRunner(output='reportes'))
72
```

Login exitoso

login_page

filled_login_form

successful_login

Crear Publicación Buscar Publicaciones

Publicaciones

Buscar:

Buscar

Título	Contenido	
Nuevo título de la publicación	Nuevo contenido de la publicación	Editar
Título de prueba	Contenido de prueba	Editar
da	sd	Editar
Título de prueba	Contenido de pruebas	Editar
Título de prueba	Contenido de prueba	Editar
Título de prueba	Contenido de prueba	Editar
Título de prueba	Contenido de prueba	Editar
Término de búsqueda	Contenido de prueba	Editar

Resultado de pueba unitaria

Resultados De La Prueba Unitaria

Hora de inicio: 2024-04-06 22:27:09

Duración: 7,29 s

Resumen: Total: 2, Aprobado: 1, Error: 1

__main__.TestLogin	Estado	
test_failed_login	Error	Vista
test_successful_login	Aprobar	

Total: 2, Pase: 1, Error: 1 -- Duración: 7,29 s

Todas las pruebas indican cual es el tipo de error y donde esta al darle clic a vista.

CS20220465