

Pruebas unitarias

ID: Tarea

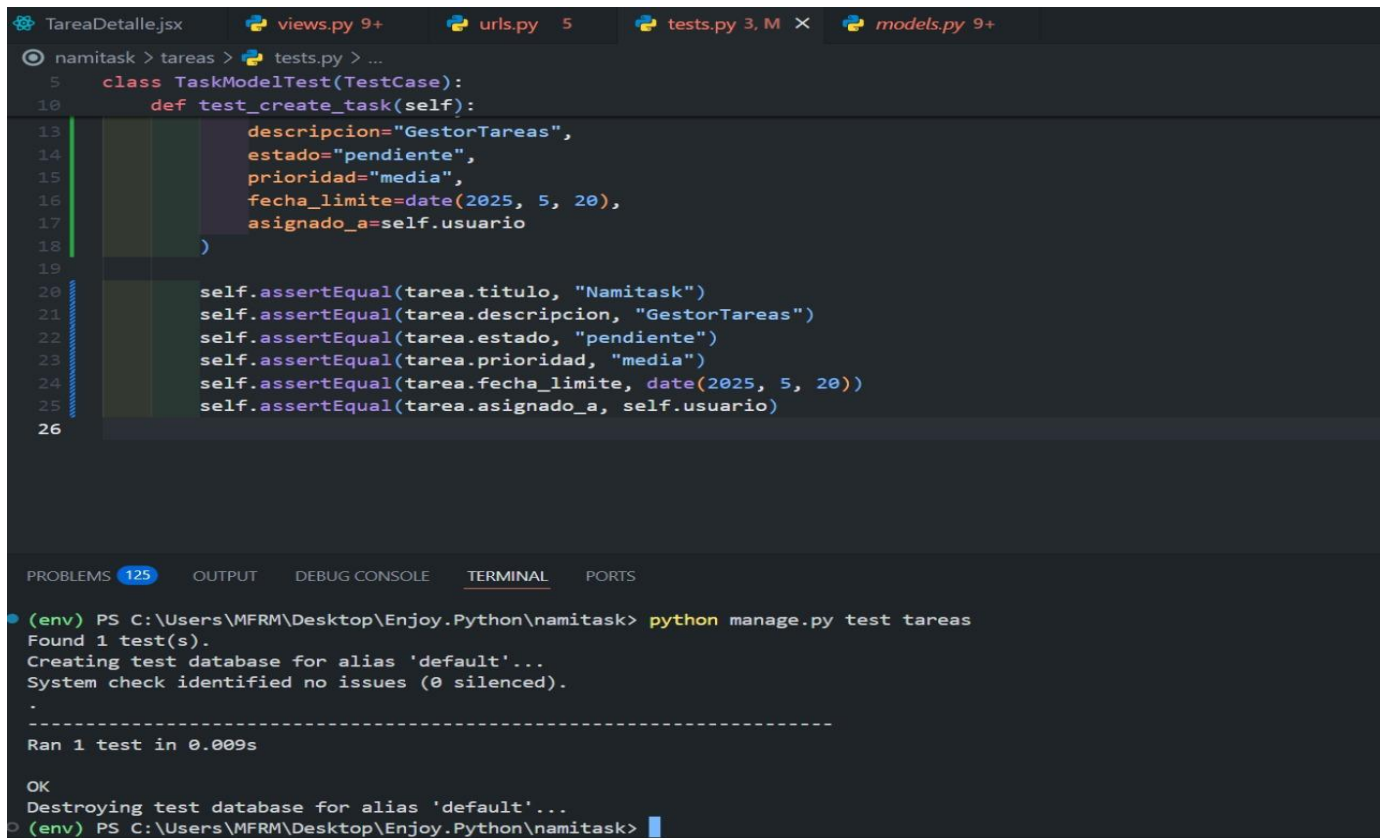
PRIORIDAD: Alta

RESUMEN: Validación de parámetros en la instancia de tarea

CÓDIGO: tarea = Tarea(title="Namitask", description="GestorTareas", completed=False, due_date="2025-05-20")

RESULTADO: Éxito

sa - La instancia de tarea se creó correctamente sin errores



```
TareaDetalle.jsx  views.py 9+  urls.py 5  tests.py 3, M x  models.py 9+
namitask > tareas > tests.py > ...
5  class TaskModelTest(TestCase):
10     def test_create_task(self):
13         descripcion="GestorTareas",
14         estado="pendiente",
15         prioridad="media",
16         fecha_limite=date(2025, 5, 20),
17         asignado_a=self.usuario
18     )
19
20     self.assertEqual(tarea.titulo, "Namitask")
21     self.assertEqual(tarea.descripcion, "GestorTareas")
22     self.assertEqual(tarea.estado, "pendiente")
23     self.assertEqual(tarea.prioridad, "media")
24     self.assertEqual(tarea.fecha_limite, date(2025, 5, 20))
25     self.assertEqual(tarea.asignado_a, self.usuario)
26

PROBLEMS 125  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
(env) PS C:\Users\MFRM\Desktop\Enjoy.Python\namitask> python manage.py test tareas
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
-----
Ran 1 test in 0.009s

OK
Destroying test database for alias 'default'...
(env) PS C:\Users\MFRM\Desktop\Enjoy.Python\namitask>
```

ID: Views **PRIORIDAD:** Alta

RESUMEN: Creación de una tarea correctamente con los parámetros adecuados en la prueba unitaria.

CODIGO: from .models import Tarea

from datetime import date

from django.test import TestCase

class TaskModelTest(TestCase):

def test_create_task(self):

tarea = Tarea.objects.create(

nombre="Namitask",

```

        descripcion="GestorTareas",
        completado=False,
        fecha_vencimiento=date(2025, 5, 20)
    )
    self.assertEqual(tarea.nombre, "Namitask")
    self.assertFalse(tarea.completado)
    self.assertEqual(tarea.fecha_vencimiento, date(2025, 5, 20))

```

RESULTADO: Ran 1 test in 0.004s

OK

Destroying test database for alias 'default'...La prueba pasó exitosamente, lo que indica que el modelo Tarea funciona correctamente con los parámetros adecuado

The screenshot shows a code editor with a file named `tests.py` open. The code defines a `TareaViewsTest` class with two test methods: `test_creacion_tarea` and `test_eliminacion_tarea`. The `test_creacion_tarea` method creates a task with specific attributes and asserts that the response status code is 201 and the task exists in the database. The `test_eliminacion_tarea` method deletes a task and asserts that the response status code is 204 and the task no longer exists in the database.

Below the code editor, the terminal window shows the output of running the tests. It indicates that 4 tests were found and run successfully in 3.488 seconds. The output also shows the creation and destruction of the test database for the alias 'default'.

```

class TareaViewsTest(TestCase):
    def test_creacion_tarea(self):
        "estado": "pendiente",
        "prioridad": "media",
        "fecha_limite": "2025-06-01",
        "asignado_a": self.usuario.id
    )
    self.assertEqual(response.status_code, 201) # Código 201 para creación exitosa
    self.assertTrue(Tarea.objects.filter(titulo="Nueva tarea").exists())

    def test_eliminacion_tarea(self):
        """Prueba la eliminación de una tarea"""
        response = self.client.delete(reverse("tarea-detail", args=[self.tarea.id]))
        self.assertEqual(response.status_code, 204) # Código 204 para eliminación exitosa
        self.assertFalse(Tarea.objects.filter(id=self.tarea.id).exists())

```

```

(env) PS C:\Users\MFRM\Desktop\Enjoy.Python\namitask> python manage.py test tareas
Found 4 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
....
-----
Ran 4 tests in 3.488s

OK
Destroying test database for alias 'default'...
(env) PS C:\Users\MFRM\Desktop\Enjoy.Python\namitask>

```

ID: Serializer **PRIORIDAD:** Alta

RESUMEN: Validación de la correcta estructuración y procesamiento de datos en el serializer al crear una tarea, asegurando que cumpla con los requisitos esperados en la prueba unitaria.

CODIGO: from django.test import TestCase
 from .serializers import TareaSerializer
 from .models import Tarea, Usuario, Etiqueta
 from datetime import date

```

class TareaSerializerTest(TestCase):
    def setUp(self):
        self.usuario = Usuario.objects.create_user(email="test@example.com", nombre_completo="Test
User", password="testpassword")
        self.etiqueta = Etiqueta.objects.create(nombre="Trabajo", color="azul")

    def test_serializer_valido(self):
        """Prueba un serializer con datos válidos"""
        datos = {
            "titulo": "Tarea de prueba",
            "descripcion": "Descripción de prueba",
            "estado": "pendiente",
            "prioridad": "media",
            "fecha_limite": "2025-05-20",
            "asignado_a": self.usuario.id,
            "etiquetas_ids": [self.etiqueta.id] #
        }
        serializer = TareaSerializer(data=datos)

        if not serializer.is_valid():
            print("Errores de validación:", serializer.errors)

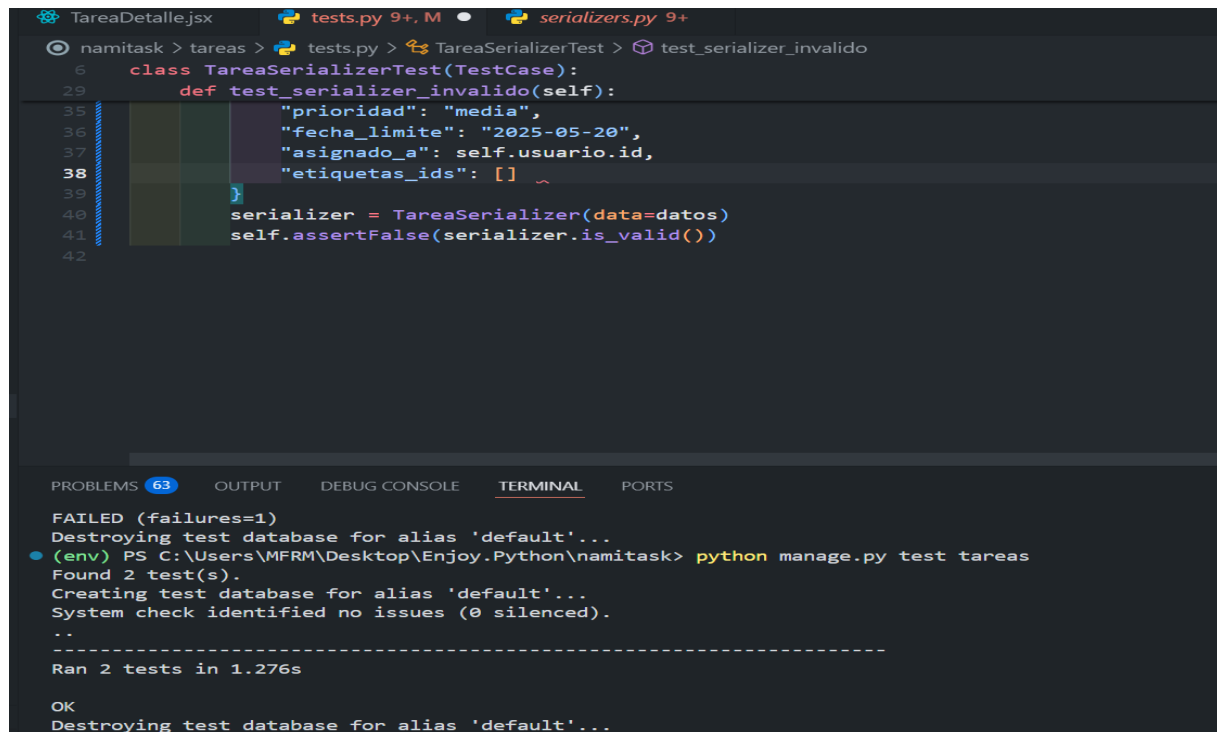
        self.assertTrue(serializer.is_valid())

    def test_serializer_invalido(self):
        """Prueba un serializer con datos inválidos"""
        datos = {
            "titulo": "", # Campo vacío
            "descripcion": "Descripción sin título",
            "estado": "pendiente",
            "prioridad": "media",
            "fecha_limite": "2025-05-20",
            "asignado_a": self.usuario.id,
            "etiquetas_ids": []
        }
        serializer = TareaSerializer(data=datos)
        self.assertFalse(serializer.is_valid())

```

RESULTADO: Ran 2 tests in 1.276s OK Destroying test database for alias 'default'...

La prueba pasó exitosamente, lo que indica que el serializer TareaSerializer valida y procesa correctamente los datos de una tarea, asegurando que cumpla con los parámetros adecuados en la prueba unitaria.



The screenshot shows a code editor with a dark theme. The top bar displays the file path: `TareaDetalle.jsx`, `tests.py 9+, M`, and `serializers.py 9+`. The editor is open to `tests.py` at line 38, showing a test method `test_serializer_invalido` for `TareaSerializerTest`. The test method sets up a `TareaSerializer` with invalid data and asserts that it is not valid. The bottom panel shows the `TERMINAL` output, indicating that the tests failed (1 failure) and providing details about the test environment and the specific test results.

```
namitask > tareas > tests.py > TareaSerializerTest > test_serializer_invalido
6 class TareaSerializerTest(TestCase):
29     def test_serializer_invalido(self):
35         "prioridad": "media",
36         "fecha_limite": "2025-05-20",
37         "asignado_a": self.usuario.id,
38         "etiquetas_ids": []
39     }
40     serializer = TareaSerializer(data=datos)
41     self.assertFalse(serializer.is_valid())
42

PROBLEMS 63 OUTPUT DEBUG CONSOLE TERMINAL PORTS
FAILED (failures=1)
Destroying test database for alias 'default'...
(env) PS C:\Users\MFRM\Desktop\Enjoy.Python\namitask> python manage.py test tareas
Found 2 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..
-----
Ran 2 tests in 1.276s

OK
Destroying test database for alias 'default'...
```

ID: Login **PRIORIDAD:** Alta

RESUMEN: Validación del correcto procesamiento y autenticación de credenciales de usuario en el sistema de login, asegurando que la autenticación funcione conforme a los requisitos esperados en la prueba unitaria.

CODIGO: from django.test import TestCase
from django.urls import reverse
from django.contrib.auth import get_user_model
Usuario = get_user_model()

```
class LoginTestCase(TestCase):
    def setUp(self):
        self.usuario = Usuario.objects.create_user(email="testuser@example.com", nombre_completo="Test User", password="testpassword")

    def test_login_exitoso(self):
        """Prueba el login con credenciales correctas"""
        response = self.client.post(reverse("token_obtain_pair"), {"email": "testuser@example.com", "password": "testpassword"})
        self.assertEqual(response.status_code, 200)
        self.assertIn("access", response.data)

    def test_login_fallido(self):
        """Prueba el login con credenciales incorrectas"""
```

```

        response = self.client.post(reverse("token_obtain_pair"), {"email": "testuser@example.com",
"password": "wrongpassword"})
        self.assertEqual(response.status_code, 401)

```

RESULTADO: Ran 2 tests in 3.318s OK Destroying test database for alias 'default'...

La prueba pasó exitosamente, lo que indica que el sistema de autenticación valida y procesa correctamente las credenciales de usuario, asegurando que el login funcione con los parámetros adecuados en la prueba unitaria.

The screenshot shows a code editor with a file named `tests.py` containing a `LoginTestCase` class. The class has a `setUp` method that creates a test user and two test methods: `test_login_exitoso` (successful login) and `test_login_fallido` (failed login). The terminal output shows the command `python manage.py test tareas` being executed, which runs the tests successfully in 3.318s.

```

1  from django.test import TestCase
2  from django.urls import reverse
3  from django.contrib.auth import get_user_model
4  Usuario = get_user_model()
5
6  class LoginTestCase(TestCase):
7      def setUp(self):
8          self.usuario = Usuario.objects.create_user(email="testuser@example.com", nombre_completo="Test User", pas
9
10
11      def test_login_exitoso(self):
12          """Prueba el login con credenciales correctas"""
13          response = self.client.post(reverse("token_obtain_pair"), {"email": "testuser@example.com", "password": "
14          self.assertEqual(response.status_code, 200)
15          self.assertIn("access", response.data)
16
17      def test_login_fallido(self):
18          """Prueba el login con credenciales incorrectas"""
19          response = self.client.post(reverse("token_obtain_pair"), {"email": "testuser@example.com", "password": "
20          self.assertEqual(response.status_code, 401)

```

```

(env) PS C:\Users\MFRM\Desktop\Enjoy.Python\namitask> python manage.py test tareas
Found 2 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..
-----
Ran 2 tests in 3.318s

OK
Destroying test database for alias 'default'...

```

ID: Registro **PRIORIDAD:** Alta

RESUMEN: Validación del correcto procesamiento de los datos de registro de usuario, incluyendo la creación exitosa con datos válidos, el manejo de contraseñas no coincidentes y la prevención de registros con correos electrónicos ya existentes, asegurando que el sistema de registro funcione conforme a los requisitos esperados en la prueba unitaria.

CODIGO: from django.test import TestCase
from django.urls import reverse
from .models import Usuario # Importa tu modelo de usuario

```

class RegistroTestCase(TestCase):
    def test_registro_exitoso(self):
        """Prueba el registro de usuario con datos válidos"""
        response = self.client.post(reverse("register"), {
            "email": "nuevo_usuario@example.com", # Usando 'email' como en tu serializer

```

```

        "password": "securepassword",
        "password2": "securepassword",
        "nombre_completo": "Nuevo Usuario" # Asegúrate de incluir todos los campos requeridos
    })
    self.assertEqual(response.status_code, 201) # Usuario registrado exitosamente

def test_registro_fallido_contrasenas_no_coincidentes(self):
    """Prueba el registro con contraseñas no coincidentes"""
    response = self.client.post(reverse("register"), {
        "email": "otro_usuario@example.com", # Usando 'email'
        "password": "securepassword",
        "password2": "wrongpassword",
        "nombre_completo": "Otro Usuario" # Incluye todos los campos requeridos
    })
    self.assertEqual(response.status_code, 400) # Error en el registro

def test_registro_fallido_email_existente(self):
    """Prueba el registro con un email ya existente"""
    # Crea un usuario existente PARA ESTA PRUEBA
    Usuario.objects.create_user(email="usuario_existente@example.com", password="password",
    nombre_completo="Usuario Existente")

    # Intenta registrar otro usuario con el mismo email
    response = self.client.post(reverse("register"), {
        "email": "usuario_existente@example.com",
        "password": "anotherpassword",
        "password2": "anotherpassword",
        "nombre_completo": "Otro Usuario"
    })
    self.assertEqual(response.status_code, 400) # Error por email existente
    self.assertIn('email', response.data) # Verifica que el error esté en el campo 'email'

```

RESULTADO: an 3 tests in 2.503s OK Destroying test database for alias 'default'...

La prueba pasó exitosamente, lo que indica que el sistema de registro valida y procesa correctamente los datos de usuario, permitiendo la creación de nuevos usuarios con información válida y rechazando intentos de registro con contraseñas incorrectas o correos electrónicos ya existentes, conforme a los parámetros definidos en la prueba unitaria.

```
tests.py 9+, M X
namitask > tareas > tests.py > RegistroTestCase > test_registro_exitoso
5 class RegistroTestCase(TestCase):
26 def test_registro_fallido_email_existente(self):
27     """Prueba el registro con un email ya existente"""
28     # Crea un usuario existente PARA ESTA PRUEBA
29     Usuario.objects.create_user(email="usuario_existente@example.com", password="password", nombre_completo="Otro Usuario")
30
31     # Intenta registrar otro usuario con el mismo email
32     response = self.client.post(reverse("register"), {
33         "email": "usuario_existente@example.com",
34         "password": "anotherpassword",
35         "password2": "anotherpassword",
36         "nombre_completo": "Otro Usuario"
37     })
38     self.assertEqual(response.status_code, 400) # Error por email existente
39     self.assertIn('email', response.data) # Verifica que el error esté en el campo 'email'

PROBLEMS 17 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Found 3 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Usuario creado: nuevo_usuario@example.com
...
-----
Ran 3 tests in 2.503s

OK
Destroying test database for alias 'default'...
```