Tests

Oscar Ivan Ulises Gutierrez Palacios
Samuel David Castaneda Mora
Juan Jose Roldan Garay
Gabriel Cuellar Garzon

Universidad Nacional de Colombia
Ingeniería de Software 1 2016701

Profesor:
Oscar Eduardo Alvarez Rodriguez

16 de noviembre 2025

Test Módulo Perfil.

1. **PerfilSecurityTest**.

   Objetivo: Validar que un usuario no pueda modificar los datos del perfil de otro usuario.

   Casos límite cubierto: Intento de modificación no autorizada: Un usuario autenticado intenta modificar el perfil de otro usuario.

   Ejecutable desde entorno: Las pruebas fueron ejecutadas desde el entorno del contenedor utilizando este comando:
   docker-compose exec web python manage.py test usuarios.tests.PerfilSecurityTest.test_usuario_no_puede_modificar_otro_perfil

2. **PerfilFormTests**.

   Objetivo: Validar que el formulario utilizado para la edición de la información del perfil de los usuarios acepte y rechace los datos adecuadamente.

   Casos límite:
   - **Edad fuera de límite:** La edad debe ser mayor o igual a 5 y menor o igual a 100
   - **Edad invalida:** Valores negativos
   - **Emails duplicados:** Si el usuario se registra o actualiza su información utilizando un correo ya existente, el test falla.

   Ejecutable desde entorno: Las pruebas fueron ejecutadas desde el entorno del contenedor utilizando este comando: docker-compose exec web python manage.py test usuarios.tests.PerfilFormTests

3. **EditarPerfilViewTests**

   Objetivo: Verificar que la vista “editar_perfil” cumpla correctamente la funcionalidad declarada

   Casos límite: Valores ingresados inválidos (Correo duplicado, edad inválida, nombre vacio)

   Ejecutable desde entorno: Las pruebas fueron ejecutadas desde el entorno del contenedor utilizando este comando: docker-compose exec web python manage.py test usuarios.tests.EditarPerfilViewTests

Test Módulo Registro.

TestsContraseña

4. **Test 1: test_contrasenas_diferentes**

Objetivo: Verificar que el método clean() del formulario Paso3SeguridadForm detecte correctamente cuando las contraseñas ingresadas en los campos password1 y password2 no coinciden.
Casos límite:

- Contraseñas válidas e iguales (caso de éxito)
- Contraseñas completamente diferentes
- Diferencia en mayúsculas/minúsculas
- Solo difieren en últimos caracteres

Ejecutable desde entorno: Las pruebas fueron ejecutadas desde el entorno del contenedor utilizando este comando:

```
Shell
docker-compose --env-file .env.dev exec web python manage.py test
usuarios.tests.test_forms.TestsContraseña.test_contrasenas_diferentes
--keepdb
```

5. **test_contrasena_tamano_minimo**

Objetivo:Verificar que el método `clean_password1()` del formulario `Paso3SeguridadForm` valide correctamente que las contraseñas cumplan con el requisito de longitud mínima de 8 caracteres.

Casos límite:

- Contraseñas con longitud menor a 8 caracteres (7, 3, 2, 1 caracteres)
- Valores en el límite inferior de validación (justo debajo del mínimo requerido)

Ejecutable desde entorno:Las pruebas fueron ejecutadas desde el entorno del contenedor utilizando este comando:

```
Shell
docker-compose --env-file .env.dev exec web python manage.py test
usuarios.tests.test_forms.TestsContraseña.test_contrasena_tamano_mini
mo --keepdb
```

TestFormularioUsuario

6. **test_clean_username**

Objetivo: Verificar que el método `clean_username()` del formulario `Paso1PersonalForm` valide correctamente que el nombre de usuario no esté duplicado en la base de datos antes de permitir el registro.

Casos límite:
- Username válido (no existe en la base de datos)
- Username duplicado (ya existe en la base de datos)
- Username vacío (campo obligatorio)

Ejecutable desde entorno: Las pruebas fueron ejecutadas desde el entorno del contenedor utilizando este comando:

```
Shell
docker-compose --env-file .env.dev exec web python manage.py test
usuarios.tests.test_forms.TestFormularioUsuario.test_clean_username --keepdb
```

7. **test_clean_email**

Objetivo:Verificar que el método `clean_email()` del formulario `Paso1PersonalForm` valide correctamente que el correo electrónico no esté duplicado en la base de datos y que tenga un formato válido antes de permitir el registro.

Casos límite:
- Email válido (no existe en la base de datos y formato correcto)
- Email duplicado (ya existe en la base de datos)
- Email con formato inválido
- Email vacío (campo obligatorio)

Ejecutable desde entorno: Las pruebas fueron ejecutadas desde el entorno del contenedor utilizando este comando:

```
Shell
docker-compose --env-file .env.dev exec web python manage.py test
usuarios.tests.test_forms.TestFormularioUsuario.test_clean_email
--keepdb
```

DashBoard Administración

TestAdminRequiredMixin

8. **test_verificar_permisos_administrador**

Objetivo: Verificar que el mixin AdminRequiredMixin valide correctamente los permisos de administrador para diferentes tipos de usuarios, incluyendo usuarios autenticados con permisos, usuarios autenticados sin permisos, y usuarios anónimos.

Casos límite:

- Usuario autenticado con permisos de administrador (debe permitir acceso)
- Usuario autenticado sin permisos de administrador (debe denegar acceso)
- Usuario no autenticado/anónimo (debe denegar acceso).

Ejecutable desde entorno: Las pruebas fueron ejecutadas desde el entorno del contenedor utilizando este comando:

```
Shell
docker-compose --env-file .env.dev exec web python manage.py test
preguntas.tests.test_views_admin.TestAdminRequiredMixin --keepdb
```

TestTemarioAddTemasView

9. **test_agregar_temas_a_componente**

Objetivo: Verificar que la vista TemarioAddTemasView agregue correctamente múltiples temas a un componente mediante bulk_create, incluyendo la validación de casos donde se agregan temas válidos individualmente o en grupo, el manejo de duplicados mediante ignore_conflicts=True, y el comportamiento cuando no se selecciona ningún tema.

Casos límite:

- Agregar un solo tema al componente (debe crear 1 registro Temario y mostrar mensaje de éxito)
- Agregar múltiples temas al componente (debe crear 2 registros Temario y mostrar mensaje de éxito)
- Agregar tema cuando ya existe otro asociado (debe crear nuevo registro sin duplicar el existente, validando bulk_create con ignore_conflicts)
- No seleccionar ningún tema (debe redirigir sin crear registros y mostrar mensaje de advertencia)

Ejecutable desde entorno:

```
Shell
docker-compose --env-file .env.dev exec web python manage.py test
preguntas.tests.test_views_admin.TestTemarioAddTemasView.test_agregar_temas_
a_componente --keepdb
```

Test Módulo Login.

10. **test_login_falla_con_email_en_vez_de_username**

    Objetivo: El usuario debe ingresar utilizando el nombre de usuario y rechazar los intentos con correo.

    Casos límite: Ingresar un correo que coincide exactamente con un username válido

    Ejecutable desde entorno: Las pruebas fueron ejecutadas desde el entorno del contenedor utilizando este comando:
    docker-compose --env-file .env.dev exec web python manage.py test usuarios.tests.LoginTests.test_login_exitoso_redireccion

11. **test_login_falla_con_password_incorrecta**

    Objetivo: El sistema sólo podrá aceptar el intento de inicio de sesión del usuario si la contraseña es correcta.

    Casos límite:
    - Contraseña incorrecta pero muy similar a la correcta
    - Contraseña incorrecta pero con caracteres especiales o Unicode
    - Contraseña vacía o sólo espacios
    - Contraseña incorrecta con caracteres de escape

    Ejecutable desde entorno: Las pruebas fueron ejecutadas desde el entorno del contenedor utilizando este comando:
    docker-compose --env-file .env.dev exec web python manage.py test usuarios.tests.LoginTests.test_login_falla_con_password_incorrecta

12. **test_login_exitoso_redireccion**

    Objetivo: Una vez el usuario haya ingresado con las credenciales correctas, el sistema debe redirigir al usuario a la página inicial.

    Casos límite:
    - URL de redirección extremadamente larga o con caracteres especiales
    - Usuario con rol especial que debe ir a una página específica
    - Redirección cuando la página de destino ya no existe (404)

    Ejecutable desde entorno: Las pruebas fueron ejecutadas desde el entorno del contenedor utilizando este comando:
    docker-compose --env-file .env.dev exec web python manage.py test usuarios.tests.LoginTests.test_login_exitoso_redireccion