Informe de ejecución de casos de prueba

- Fecha: 21/12/2023
- Autores: Viscomi Leandro
- Sistema: Windows, Django
- Versión del sistema: Windows 11 y Django 3.2

Prueba 1 “Pagina tipo Blog”.

CP ID	**Caso documentado 01**	
Sistema:	Tipo Prueba: TestCase de Django 3.2	
	Ambiente de Prueba: Dentro de tests.py funcionando como Test	
	Autor: Viscomi Leandro	
	Responsable ejecución: manage.py	
	Fecha Creación: 21/12/2023	Fecha Ejecución: 21/12/2023

Datos de Prueba (Entrada):

```
class PaginaTestCase(TestCase):
    def setUp(self):
        # Configuración común para las pruebas
        self.user = User.objects.create_user(username='testuser', password='testpassword')
        self.profesor = Profesores.objects.create(nombre='Test', apellido='Profesor', email='test@example.com',
profesion='TestProf')
        self.curso = Curso.objects.create(nombre='Test Curso', comision=101)

    def test_acceso_vistas_protegidas(self):
        # Prueba de acceso a vistas protegidas sin inicio de sesión
        response = self.client.get(reverse('Bienvenida'))
        self.assertEqual(response.status_code, 302, "Se esperaba una redirección a la página de inicio de sesión")

        # Inicia sesión y prueba el acceso nuevamente
        self.client.login(username='testuser', password='testpassword')
        response = self.client.get(reverse('Bienvenida'))
        self.assertEqual(response.status_code, 200, "Se esperaba acceso permitido")

    def test_creacion_profesor(self):
        # Prueba la creación de un profesor
        self.client.login(username='testuser', password='testpassword')
        response = self.client.post(reverse('NuevoProfesor'), {'nombre': 'Nuevo', 'apellido': 'Profesor', 'email':
'nuevo@example.com', 'profesion': 'NuevoProf'}, follow=True)
        self.assertEqual(response.status_code, 200, "La creación del profesor no devolvió el código de estado esperado")

        # Verifica que el profesor se haya creado correctamente
        nuevo_profesor = Profesores.objects.get(email='nuevo@example.com')
        self.assertEqual(nuevo_profesor.nombre, 'Nuevo')
        self.assertEqual(nuevo_profesor.apellido, 'Profesor')
```

```
    def test_creacion_curso(self):
        # Prueba la creación de un curso
        self.client.login(username='testuser', password='testpassword')
        response = self.client.post(reverse('NuevoCurso'), {'nombre': 'Nuevo Curso', 'comision': 102}, follow=True)
        self.assertEqual(response.status_code, 200, "La creación del curso no devolvió el código de estado esperado")

        # Verifica que el curso se haya creado correctamente
        nuevo_curso = Curso.objects.get(nombre='Nuevo Curso')
        self.assertEqual(nuevo_curso.comision, 102)
```

Precondiciones para ejecución:

```
from django.test import TestCase
from django.contrib.auth.models import User
from django.urls import reverse
from Pagina.models import Profesores, Curso
```

Ejecución: APROBADA

Informe de ejecución de casos de prueba

- Fecha: 21/12/2023
- Autores: Viscomi Leandro
- Sistema: Windows, Django
- Versión del sistema: Windows 11 y Django 3.2

Prueba 2 “Pagina tipo Blog”.

CP ID	**Caso documentado 02**	
Sistema:	Tipo Prueba: TestCase de Django 3.2	
	Ambiente de Prueba: Dentro de tests.py funcionando como Test	
	Autor: Viscomi Leandro	
	Responsable ejecución: manage.py	
	Fecha Creación: 21/12/2023	Fecha Ejecución: 21/12/2023

Datos de Prueba (Entrada):

```
class PaginaTestCase(TestCase):
    def setUp(self):
        self.user = User.objects.create_user(username='testuser', password='testpassword')
        self.profesor = Profesores.objects.create(nombre='Test', apellido='Profesor', email='test@example.com',
profesion='TestProf')
        self.curso = Curso.objects.create(nombre='Test Curso', comision=101)
        self.estudiante = Estudiantes.objects.create(nombre='Test', apellido='Estudiante', email='test@example.com')
        self.entregable = Entregable.objects.create(nombre='Test Entregable', fecha_entrega='2023-12-31', entregado='S')

    def test_acceso_vistas_protegidas(self):
        response = self.client.get(reverse('Bienvenida'))
        self.assertEqual(response.status_code, 302)  # Redirección a la página de inicio de sesión

        self.client.login(username='testuser', password='testpassword')
        response = self.client.get(reverse('Bienvenida'))
        self.assertEqual(response.status_code, 200)  # Acceso permitido

    def test_creacion_profesor(self):
        self.client.login(username='testuser', password='testpassword')
        response = self.client.post(reverse('NuevoProfesor'), {'nombre': 'Nuevo', 'apellido': 'Profesor', 'email':
'nuevo@example.com', 'profesion': 'NuevoProf'})
        self.assertEqual(response.status_code, 302)  # Cambiado de 200 a 302, ya que es una redirección

        nuevo_profesor = Profesores.objects.get(email='nuevo@example.com')
        self.assertEqual(nuevo_profesor.nombre, 'Nuevo')
        self.assertEqual(nuevo_profesor.apellido, 'Profesor')

    def test_creacion_curso(self):
        self.client.login(username='testuser', password='testpassword')
        response = self.client.post(reverse('NuevoCurso'), {'nombre': 'Nuevo Curso', 'comision': 102})
```

```
        self.assertEqual(response.status_code, 302)  # Cambiado de 200 a 302

        response = self.client.get(response.url)
        self.assertEqual(response.status_code, 200)

        nuevo_curso = Curso.objects.get(nombre='Nuevo Curso')
        self.assertEqual(nuevo_curso.comision, 102)

    def test_creacion_estudiante(self):
        self.client.login(username='testuser', password='testpassword')
        response = self.client.post(reverse('NuevoEstudiante'), {'nombre': 'Nuevo', 'apellido': 'Estudiante', 'email':
'nuevo_estudiante@example.com'})
        self.assertEqual(response.status_code, 302)  # Cambiado de 200 a 302

        nuevo_estudiante = Estudiantes.objects.get(email='nuevo_estudiante@example.com')
        self.assertEqual(nuevo_estudiante.nombre, 'Nuevo')
        self.assertEqual(nuevo_estudiante.apellido, 'Estudiante')

    def test_creacion_entregable(self):
        self.client.login(username='testuser', password='testpassword')
        response = self.client.post(reverse('NuevoEntregable'), {'nombre': 'Nuevo Entregable', 'fecha_entrega': '2023-12-
31', 'entregado': 'S'})
        self.assertEqual(response.status_code, 302)  # Cambiado de 200 a 302

        nuevo_entregable = Entregable.objects.get(nombre='Nuevo Entregable')
        self.assertEqual(nuevo_entregable.fecha_entrega.strftime('%Y-%m-%d'), '2023-12-31')
        self.assertEqual(nuevo_entregable.entregado, 'S')
```

Precondiciones para ejecución:

```
from django.test import TestCase
from django.contrib.auth.models import User
from django.urls import reverse
from Pagina.models import Profesores, Curso, Estudiantes, Entregable
```

Ejecución: APROBADA

Informe de ejecución de casos de prueba

- Fecha: 21/12/2023
- Autores: Viscomi Leandro
- Sistema: Windows, Django
- Versión del sistema: Windows 11 y Django 3.2

Prueba 3 “Pagina tipo Blog”.

CP ID	**Caso documentado 02**	
Sistema:	Tipo Prueba: TestCase de Django 3.2	
	Ambiente de Prueba: Dentro de tests.py funcionando como Test	
	Autor: Viscomi Leandro	
	Responsable ejecución: manage.py	
	Fecha Creación: 21/12/2023	Fecha Ejecución: 21/12/2023
Datos de Prueba (Entrada):		

```
class PaginaTestCase(TestCase):
    def setUp(self):
        # Configuración común para las pruebas
        self.user = User.objects.create_user(username='testuser', password='testpassword', email='test@example.com')
        self.avatar = Avatar.objects.create(user=self.user, imagen='path/to/test/image.jpg')

    def test_edicion_perfil(self):
        # Inicia sesión
        self.client.login(username='testuser', password='testpassword')

        # Accede a la vista EditarPerfil
        response = self.client.get(reverse('EditarPerfil'))
        self.assertEqual(response.status_code, 200)  # Verifica que se pueda acceder a la vista

        # Realiza una edición en el perfil (ajusta los datos según tus necesidades)
        response = self.client.post(
            reverse('EditarPerfil'),
            {'username': 'testuser', 'first_name': 'NuevoNombre', 'last_name': 'NuevoApellido', 'email': 'test@example.com'}
        )

        # Verifica que la edición fue exitosa (código 302 significa redirección)
        self.assertIn(response.status_code, [200, 302])

        # Si hay redirección, sigue la redirección
        if response.status_code == 302:
            response = self.client.get(response.url)
            self.assertEqual(response.status_code, 200)  # Verifica el código después de la redirección

        # Obtiene el usuario actualizado desde la base de datos
        user_actualizado = User.objects.get(username='testuser')
```

```
    # Verifica que los cambios se hayan aplicado correctamente
    self.assertEqual(user_actualizado.first_name, 'NuevoNombre')
    self.assertEqual(user_actualizado.last_name, 'NuevoApellido')

    # Verifica que la imagen del avatar no se haya modificado (opcional)
    avatar_actualizado = Avatar.objects.get(user=user_actualizado)
            self.assertEqual(avatar_actualizado.imagen, 'path/to/test/image.jpg')
```

Precondiciones para ejecución:

```
from django.test import TestCase
from django.contrib.auth.models import User
from django.urls import reverse
from Pagina.models import Avatar
```

Ejecución: APROBADA