Informe de ejecución de casos de prueba

- Fecha: 30/04/23
- Autores: Rosas Sandillú, Julián
- Sistema: Windows, Django
- Versión del sistema: Windows 10 y Django 3.2

CP-01 “CRUD de Videojuego”.

CP ID	**Caso documentado 01**	
Nombre Caso de Prueba:	“CRUD de Videojuego”	
Descripción:	Un test para ver la funcionalidad del CRUD de Videojuego llevado al fallo	
Versión:	1.0	
Prioridad	Baja, ya se había comprobado casi por completo su funcionalidad	
Sistema:	Nivel Prueba: Avanzado, llevado al fallo	
	Tipo Prueba: TestCase de Django 3.2	
	Ambiente de Prueba: Dentro de tests.py funcionando como Test	
	Autor: Rosas Sandillú, Julián	
	Responsable ejecución: manage.py	
	Fecha Creación: 30/04/23	Fecha Ejecución: 30/04/23

Datos de Prueba (Entrada):

```
class VideojuegoTestCase(TestCase):

    def setUp(self):
        self.client = Client()
        self.user = User.objects.create_user('testuser', 'test@example.com', 'testpassword')
        self.videojuego = Videojuego.objects.create(nombre='Juego de Prueba', fecha_salida='2022-01-01',
genero='Aventura', empresa='DesarrolladorX', descripcion='Un juego de prueba', valoracion=4)

    def test_agregar_videojuego(self):
        self.client.login(username='testuser', password='testpassword')
        url = reverse('agregar_videojuego')
        data = {'nombre': 'Nuevo Juego', 'fecha_salida': '2023-04-30', 'genero': 'Estrategia', 'empresa': 'DesarrolladorY',
'descripcion': 'Un juego nuevo', 'valoracion': 3}
        response = self.client.post(url, data)
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, "Muy bien agregaste tu videojuego")

    def test_lista_videojuegos(self):
        self.client.login(username='testuser', password='testpassword')
        url = reverse('lista_videojuegos')
```

```
        response = self.client.get(url)
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, "Juego de Prueba")

    def test_eliminar_videojuego(self):
        self.client.login(username='testuser', password='testpassword')
        url = reverse('eliminar_videojuego', args=[self.videojuego.id])
        response = self.client.post(url)
        self.assertEqual(response.status_code, 302)
        self.assertRedirects(response, '/l-videojuegos')

    def test_editar_videojuego(self):
        self.client.login(username='testuser', password='testpassword')
        url = reverse('editar_videojuego', args=[self.videojuego.id])
        data = {'nombre': 'Juego Modificado', 'fecha_salida': '2022-06-01', 'genero': 'Aventura', 'empresa': 'DesarrolladorX',
'descripcion': 'Un juego modificado', 'valoracion': 5}
        response = self.client.post(url, data)
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, "Cambios realizados")
```

Precondiciones para ejecución:

```
from django.test import TestCase, Client
from django.urls import reverse
from django.contrib.auth.models import User
from .models import *
from .forms import *
from .views import *
```

Ejecución:

Paso	Condición	Valor	Resultado Esperado	Resultado Obtenido
Correcto, crea todos los nombres de ejemplo.	Responde a todas las condiciones dadas y sigue hasta que no encuentre ese videojuego creado.	Responde a los valores asignados y sigue hasta que no encuentre un valor asignado.	Que funcione todo normal hasta que llegue al fallo por no encontrar un item lista_videojuegos.	El esperado.

Criterios de Aprobación (precondiciones):

- Cumplimiento del 100 % de los resultados esperados.

Decisión de Aprobación: APROBADO

Informe de ejecución de casos de prueba

- Fecha: 30/04/23
- Autores: Rosas Sandillú, Julián
- Sistema: Windows, Django
- Versión del sistema: Windows 10 y Django 3.2

CP-02 “CRUD de Reseña”.

CP ID	**Caso documentado 02**	
Nombre Caso de Prueba:	“CRUD de Reseña”	
Descripción:	Un test para ver la funcionalidad del CRUD de Reseña llevado al fallo	
Versión:	1.0	
Prioridad	Baja, ya se había comprobado casi por completo su funcionalidad	
Sistema:	Nivel Prueba: Avanzado, llevado al fallo	
	Tipo Prueba: TestCase de Django 3.2	
	Ambiente de Prueba: Dentro de tests.py funcionando como Test	
	Autor: Rosas Sandillú, Julián	
	Responsable ejecución: manage.py	
	Fecha Creación: 30/04/23	Fecha Ejecución: 30/04/23

Datos de Prueba (Entrada):

```
class TestResenasViews(TestCase):

    def setUp(self):
        self.client = Client()
        self.lista_resenas_url = reverse('lista_resenas')
        self.agregar_resena_url = reverse('agregar_resena')
        self.eliminar_resena_url = reverse('eliminar_resena', args=[1])
        self.editar_resena_url = reverse('editar_resena', args=[1])
        self.resena = Resena.objects.create(
            autor="Jhon Doe",
            contenido="Esta es una reseña de prueba",
            videojuego="Call of Duty"
        )
        self.resena_form_data = {
            "autor": "Jane Smith",
            "contenido": "Esta es otra reseña de prueba",
            "videojuego": "Fortnite"
        }

    def test_lista_resenas_GET(self):
        response = self.client.get(self.lista_resenas_url)
```

```
        self.assertEquals(response.status_code, 200)
        self.assertTemplateUsed(response, 'lista_resenas.html')

    def test_agregar_resena_POST(self):
        response = self.client.post(self.agregar_resena_url, self.resena_form_data)

        self.assertEquals(response.status_code, 302)
        self.assertEquals(Resena.objects.last().autor, 'Jane Smith')
        self.assertEquals(Resena.objects.last().contenido, 'Esta es otra reseña de prueba')
        self.assertEquals(Resena.objects.last().videojuego, 'Fortnite')

    def test_eliminar_resena_POST(self):
        response = self.client.post(self.eliminar_resena_url)

        self.assertEquals(response.status_code, 302)
        self.assertEquals(Resena.objects.count(), 0)

    def test_editar_resena_GET(self):
        response = self.client.get(self.editar_resena_url)

        self.assertEquals(response.status_code, 200)
        self.assertTemplateUsed(response, 'editar_resena.html')

    def test_editar_resena_POST(self):
        response = self.client.post(self.editar_resena_url, self.resena_form_data)

        self.assertEquals(response.status_code, 200)
        self.assertEquals(Resena.objects.first().autor, 'Jane Smith')
        self.assertEquals(Resena.objects.first().contenido, 'Esta es otra reseña de prueba')
        self.assertEquals(Resena.objects.first().videojuego, 'Fortnite')
```

Precondiciones para ejecución:

```
from django.test import TestCase, Client
from django.urls import reverse
from django.contrib.auth.models import User
from .models import *
from .forms import *
from .views import *
```

Ejecución:

Paso	Condición	Valor	Resultado Esperado	Resultado Obtenido
Correcto, crea todos los nombres de ejemplo.	Responde a todas las condiciones dadas y sigue hasta que no encuentre esa reseña creada.	Responde a los valores asignados y sigue hasta que no encuentre un valor asignado.	Que funcione todo normal hasta que llegue al fallo por no encontrar un item lista_resenas.	El esperado.

Criterios de Aprobación (precondiciones):

- Cumplimiento del 100 % de los resultados esperados.

Decisión de Aprobación: APROBADO

Informe de ejecución de casos de prueba

- Fecha: 30/04/23
- Autores: Rosas Sandillú, Julián
- Sistema: Windows, Django
- Versión del sistema: Windows 10 y Django 3.2

CP-03 “login de usuarios”.

<table>
<tr><td>CP ID</td><td colspan="2">Caso documentado 03</td></tr>
<tr><td>Nombre Caso de Prueba:</td><td colspan="2">“login de usuarios”</td></tr>
<tr><td>Descripción:</td><td colspan="2">Un test para ver la funcionalidad del login de usuarios llevado al fallo</td></tr>
<tr><td>Versión:</td><td colspan="2">1.0</td></tr>
<tr><td>Prioridad</td><td colspan="2">Baja, ya se había comprobado casi por completo su funcionalidad</td></tr>
<tr><td>Sistema:</td><td colspan="2">Nivel Prueba: Avanzado, llevado al fallo</td></tr>
<tr><td></td><td colspan="2">Tipo Prueba: TestCase de Django 3.2</td></tr>
<tr><td></td><td colspan="2">Ambiente de Prueba: Dentro de tests.py funcionando como Test</td></tr>
<tr><td></td><td colspan="2">Autor: Rosas Sandillú, Julián</td></tr>
<tr><td></td><td colspan="2">Responsable ejecución: manage.py</td></tr>
<tr><td></td><td>Fecha Creación: 30/04/23</td><td>Fecha Ejecución: 30/04/23</td></tr>
<tr><td colspan="3">Datos de Prueba (Entrada):</td></tr>
<tr><td colspan="3">class TestLogin(TestCase):
  def setUp(self):
  self.client = Client()
  self.login_url = reverse('login_usuarios')
  self.username = 'testuser'
  self.email = 'testuser@test.com'
  self.password = 'testpass'
  self.user = User.objects.create_user(
  username=self.username,
  email=self.email,
  password=self.password
)
def test_login(self):
  response = self.client.post(self.login_url, {'username': self.username, 'password': self.password})
  self.assertEqual(response.status_code, 302)
  self.assertRedirects(response, reverse('inicio'))
  self.assertTrue(response.wsgi_request.user.is_authenticated)</td></tr>
<tr><td colspan="3">Precondiciones para ejecución:</td></tr>
</table>

from django.test import TestCase, Client
from django.urls import reverse
from django.contrib.auth.models import User
from .models import *
from .forms import *
from .views import *

Ejecución:

Paso	Condición	Valor	Resultado Esperado	Resultado Obtenido
Correcto, crea todos los nombres de ejemplo.	Responde a todas las condiciones dadas y sigue hasta que no pueda loguearse por los permisos del server.	Responde a los valores asignados y sigue hasta que no encuentre una forma de loguearse con el server por los permisos del server.	Que funcione todo normal hasta que llegue un fallo por los permisos del server, si es que llega.	El esperado.

Criterios de Aprobación (precondiciones):

- Cumplimiento del 100 % de los resultados esperados.

Decisión de Aprobación: APROBADO