# 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”.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CP ID** | | | **Caso documentado 03** | | | |
| Nombre Caso de Prueba: | | | “login de usuarios” | | | |
| Descripción: | | | Un test para ver la funcionalidad del login de usuarios 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 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) | | | | | | |
| 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 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 | | | | | | |