

# Documentación del Módulo de Testing

## ÁgoraUN - Entrega 06

### 1. Resumen Ejecutivo

**Proyecto:** ÁgoraUN - Plataforma de Gestión de Grupos Estudiantiles

**Módulo:** Módulo de Testing (Entrega 06)

**Fecha:** Noviembre 10, 2025

**Herramienta:** unittest (Django TestCase)

**Total de Pruebas:** 12

**Estado:**  COMPLETADO - 12/12 tests PASADOS

**Tiempo de Ejecución:** 0.012 segundos

### 2. Resumen de Pruebas Implementadas

#	ID Test	Nombre	Requerimiento	Funcionalidad	Estado
1	Test.001	test_crear_grupo_campos_requeridos	RF_1	Crear grupo con datos completos	<input checked="" type="radio"/> PASS
2	Test.002	test_validar_formato_correo_institucional	RF_1	Validación de correo @unal.edu.co	<input checked="" type="radio"/> PASS
3	Test.003	test.persistencia_datos_en_bd	RF_1	Verificar persistencia en BD	<input checked="" type="radio"/> PASS
4	Test.004	test_listar.todos.grupos.catalogo	RF_3	Listar todos los grupos	<input checked="" type="radio"/> PASS
5	Test.005	test_buscar.grupo.por.nombre	RF_3	Búsqueda por nombre (parcial)	<input checked="" type="radio"/> PASS
6	Test.006	test_filtrar.grupos.por.area.interes	RF_3	Filtrar por área de interés	<input checked="" type="radio"/> PASS
7	Test.007	test_actualizar.informacion.grupo	RF_5	Editar información (UPDATE)	<input checked="" type="radio"/> PASS
8	Test.008	test_eliminar.grupo	RF_5	Eliminar grupo (DELETE)	<input checked="" type="radio"/> PASS

#	ID Test	Nombre	Requerimiento	Funcionalidad	Estado
9	Test.009	test_ediciones_sucesivas_integridad	RF_5	Integridad con múltiples ediciones	O PASS
10	Test.010	test_crear_multiples_grupos_sin_conflicto	Validación	Crear 5 grupos sin conflictos de ID	O PASS
11	Test.011	test_validar_tipos_grupo_correctos	Validación	Validar 4 tipos diferentes	O PASS
12	Test.012	test_campos_no_nulos_en_consultas	Validación	Verificar NOT NULL en campos críticos	O PASS

### 3. Matriz de Trazabilidad: Requerimientos ↔ Tests

#### RF\_1: Registro de clubes con información completa

Requisito	Test ID	Test Name	Validación
RF_1.1: Crear grupo con datos completos	Test.001	test_crear_grupo_campos_requeridos	Todos los campos creados y almacenados
RF_1.2: Validación de correo institucional	Test.002	test_validar_formato_correo_institucional	Solo correos @unal.edu.co
RF_1.3: Persistencia en BD	Test.003	test_persistencia_datos_en_bd	Datos recuperables después de crear

#### RF\_3: Catálogo de clubes para consulta

Requisito	Test ID	Test Name	Validación
RF_3.1: Listar todos los grupos	Test.004	test_listar.todos.grupos.catalogo	Lista completa de grupos
RF_3.2: Buscar por nombre	Test.005	test_buscar.grupo_por_nombre	Búsqueda parcial funciona
RF_3.3: Filtrar por área	Test.006	test_filtrar.grupos_por_area_interes	Filtrado múltiple correcto

## RF\_5: Gestión de grupos (CRUD)

Requisito	Test ID	Test Name	Validación
RF_5.1: Editar información	Test.007	test_actualizar_informacion_grupo	UPDATE funciona
RF_5.2: Eliminar grupo	Test.008	test_eliminar_grupo	DELETE funciona
RF_5.3: Integridad de datos	Test.009	test_ediciones_sucesivas_integridad	Múltiples cambios se preservan

## Validación Integral (Casos Límite)

Requisito	Test ID	Test Name	Validación
Límite: IDs únicos en creación múltiple	Test.010	test_crear_multiples_grupos_sin_conflicto	5 grupos sin conflictos
Límite: Tipos válidos	Test.011	test_validar_tipos_grupo_correctos	4 tipos diferentes guardados
Límite: Campos no nulos	Test.012	test_campos_no_nulos_en_consultas	Integridad referencial

## 4. Descripción Detallada de Pruebas

### Test.001 - Crear grupo con campos requeridos

**Requerimiento:** RF\_1.1

**Objetivo:** Validar creación completa de grupo

**Entrada:** Datos: nombre\_grupo, descripcion, tipo\_grupo, area\_interes, correo\_grupo

**Salida Esperada:** Grupo creado con ID único

**Casos Límite:**

- Todos los campos obligatorios presentes
- Correo con formato válido
- Descripción no vacía
- ID auto-generado correctamente

**Validaciones:**

- ✓ ID generado automáticamente
- ✓ Nombre almacenado correctamente
- ✓ Correo en formato @unal.edu.co

## **Test.002 - Validación de correo institucional**

**Requerimiento:** RF\_1.2

**Objetivo:** Asegurar que solo se usen correos @unal.edu.co

**Entrada:** Correo: investigacion@unal.edu.co

**Salida Esperada:** Correo almacenado correctamente

**Casos Límite:**

- Validación obligatoria de dominio @unal.edu.co
- Longitud mínima de correo
- Formato válido de email

**Validaciones:**

- ✓ Contiene "@" obligatorio
- ✓ Dominio "unal.edu.co" presente
- ✓ No hay caracteres inválidos

## **Test.003 - Persistencia de datos en BD**

**Requerimiento:** RF\_1.3

**Objetivo:** Verificar que datos se guardan y recuperan correctamente

**Entrada:** Grupo creado con datos específicos

**Salida Esperada:** Datos recuperables mediante query

**Casos Límite:**

- Consulta posterior a creación
- Múltiples campos verificados
- Integridad de datos preservada

**Validaciones:**

- ✓ Grupo creado tiene ID único
- ✓ Datos recuperables con get()
- ✓ Todos los campos preservados

## **Test.004 - Listar todos los grupos en catálogo**

**Requerimiento:** RF\_3.1

**Objetivo:** Verificar que el catálogo lista todos los grupos

**Entrada:** 3 grupos en base de datos

**Salida Esperada:** 3 grupos retornados

**Casos Límite:**

- Conteo exacto de grupos
- Verificación de existencia de cada grupo

- Integridad de lista

#### **Validaciones:**

- ✓ Conteo exacto: 3 grupos
- ✓ Cada grupo existe en lista
- ✓ Orden consistente

### **Test.005 - Búsqueda de grupo por nombre**

**Requerimiento:** RF\_3.2

**Objetivo:** Validar funcionalidad de búsqueda

**Entrada:** Texto de búsqueda "Literatura"

**Salida Esperada:** Club de Literatura encontrado

#### **Casos Límite:**

- Búsqueda insensible a mayúsculas
- Búsqueda parcial funciona
- Campo correcto retornado

#### **Validaciones:**

- ✓ Encuentra "Club de Literatura"
- ✓ Búsqueda case-insensitive
- ✓ Retorna objeto correcto

### **Test.006 - Filtrar grupos por área de interés**

**Requerimiento:** RF\_3.3

**Objetivo:** Validar filtrado avanzado

**Entrada:** Filtro: area\_interes = "Deporte"

**Salida Esperada:** Solo 1 grupo deportivo

#### **Casos Límite:**

- Filtrado exacto funciona
- No retorna grupos de otras áreas
- Conteo específico verificado

#### **Validaciones:**

- ✓ Retorna solo grupos deportivos (1)
- ✓ Excluye otros tipos
- ✓ Conteo exacto

## **Test.007 - Actualizar información del grupo**

**Requerimiento:** RF\_5.1

**Objetivo:** Validar UPDATE de grupos

**Entrada:** Cambios en descripción y tipo\_grupo

**Salida Esperada:** Cambios persistidos en BD

**Casos Límite:**

- Múltiples campos actualizados simultáneamente
- Cambios guardados correctamente
- Datos recuperables después de save()

**Validaciones:**

- ✓ Descripción actualizada
- ✓ Tipo de grupo modificado
- ✓ Cambios persistidos

## **Test.008 - Eliminar grupo**

**Requerimiento:** RF\_5.2

**Objetivo:** Validar DELETE de grupos

**Entrada:** ID de grupo válido

**Salida Esperada:** Grupo eliminado de BD

**Casos Límite:**

- Verificación de no existencia post-delete
- Excepción DoesNotExist lanzada correctamente
- Sin datos huérfanos

**Validaciones:**

- ✓ Grupo eliminado de BD
- ✓ DoesNotExist exception lanzada
- ✓ Sin datos inconsistentes

## **Test.009 - Ediciones sucesivas con integridad**

**Requerimiento:** RF\_5.3

**Objetivo:** Validar integridad con múltiples ediciones

**Entrada:** 3 ediciones secuenciales

**Salida Esperada:** Todos los cambios preservados

**Casos Límite:**

- Edición 1: descripción
- Edición 2: area\_interes

- Edición 3: correo\_grupo

**Validaciones:**

- ✓ Primera edición: descripción actualizada
- ✓ Segunda edición: área modificada
- ✓ Tercera edición: correo cambiado
- ✓ Todos los cambios recuperables

### **Test.010 - Crear múltiples grupos sin conflicto**

**Objetivo:** Validar creación masiva sin conflictos

**Entrada:** 5 grupos creados secuencialmente

**Salida Esperada:** 5 IDs únicos, sin duplicados

**Casos Límite:**

- IDs secuenciales y únicos
- No hay colisiones
- Conteo exacto verificado

**Validaciones:**

- ✓ 5 IDs únicos generados
- ✓ No hay duplicados en IDs
- ✓ Conteo total: 5 grupos

### **Test.011 - Validar tipos de grupo correctos**

**Objetivo:** Verificar que tipos válidos se guardan

**Entrada:** 4 tipos: Académico, Investigación, Deportivo, Cultural

**Salida Esperada:** Todos los tipos almacenados correctamente

**Casos Límite:**

- Múltiples tipos diferentes
- Cada tipo verificado individualmente
- Sin corrupción de datos

**Validaciones:**

- ✓ Tipo "Académico" almacenado
- ✓ Tipo "Investigación" almacenado
- ✓ Tipo "Deportivo" almacenado
- ✓ Tipo "Cultural" almacenado

## Test.012 - Campos no nulos en consultas

**Objetivo:** Validar integridad referencial

**Entrada:** Grupo completo creado

**Salida Esperada:** Ningún campo esencial es nulo

**Casos Límite:**

- 5 campos críticos verificados
- NOT NULL enforcement
- Integridad de datos garantizada

**Validaciones:**

- ✓ nombre\_grupo NOT NULL
- ✓ descripcion NOT NULL
- ✓ correo\_grupo NOT NULL
- ✓ tipo\_grupo NOT NULL
- ✓ area\_interes NOT NULL

## 5. Ejecución de Pruebas

**Ejecutar todos los tests:**

```
docker-compose exec web python manage.py test grupos
```

**Resultado esperado:**

```
Found 12 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
.
.
Ran 12 tests in 0.012s

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

**Detalladamente:**

```
docker-compose exec web python manage.py test grupos -v 2
```

## 6. Cobertura de Funcionalidades

Funcionalidad	Requerimiento	Tests	Cobertura
Crear grupo	RF_1	3	100%
Validar datos	RF_1	1	100%
Persistencia BD	RF_1	1	100%
Listar catálogo	RF_3	1	100%
Buscar grupo	RF_3	1	100%
Filtrar grupo	RF_3	1	100%
Editar grupo	RF_5	1	100%
Eliminar grupo	RF_5	1	100%
Integridad datos	RF_5	1	100%
Validación IDs	-	1	100%
Validación tipos	-	1	100%
Validación nulos	-	1	100%
**Total: 12 Pruebas		100% de Cobertura**	

## 7. Especificaciones Técnicas

**Framework:** Django TestCase

**Herramienta:** unittest (Python estándar)

**Base de Datos:** SQLite (en memoria para tests)

**Entorno:** Docker + docker-compose

### Requisitos cumplidos:

#### ○ Herramienta adecuada al lenguaje

- unittest: Framework nativo de Python
- Django TestCase: Optimizado para aplicaciones Django

#### ○ Funcionalidad esencial del sistema

- Todas las operaciones CRUD
- Búsqueda y filtrado
- Validación de datos
- Integridad referencial

#### ○ Casos límite documentados

- Cada test especifica entrada, salida y límites

- Validaciones explícitas en código

- **Ejecutables desde el entorno del proyecto**

- Docker containerizado
- Comandos simples y reproducibles