Universidad Nacional de Colombia - sede Bogotá
Facultad de Ingeniería
Departamento de Sistemas e Industrial
Curso: Ingeniería de Software 1 (2016701)
Grupo: BIOS
Integrantes:
Oscar Daniel Aguas Quiroz,
Juan David Peña Lozada,
Rodrigo Rivera Rivera y Andres Malaver

1. INTRODUCCIÓN

Este documento describe las pruebas unitarias implementadas para los módulos de autenticación (accounts) y gestión de animales (animals) del proyecto BIOS usando pytest.

BIOS es una aplicación Django para gestionar lotes de animales donde los usuarios se registran, crean lotes y agregan animales para su seguimiento.

🛠️ Framework utilizado:

🧪 pytest: Testing moderno para Python

🔗 pytest-django: Integración con Django

💾 BD SQLite en memoria para tests rápidos

📁 2. UBICACIÓN Y EJECUCIÓN

📂 ESTRUCTURA:

Proyecto/

├── tests/

│ ├── __init__.py

│ ├── conftest.py (⚙️ configuración y fixtures)

│ ├── test_accounts.py

```
│   └── test_animals.py
├── pytest.ini (⚙️ configuración)
└── requirements.txt
```

COMANDOS:

Todos los tests

pytest

Tests específicos

pytest tests/test_accounts.py -v

pytest tests/test_animals.py -v

Con cobertura

pytest --cov=accounts --cov=animals tests/

🔍 Ver detalles de fallos

pytest -v

pytest -s (mostrar prints)

3. TESTS DE ACCOUNTS (Autenticación) - 8 TESTS

Módulo: tests/test_accounts.py

🔑 CLASE: TestEmailAuthenticationForm

✅ TEST 1: test_valid_login_with_correct_credentials

❓ ¿Por qué?: Validar que usuarios puedan loguearse con credenciales válidas

Prueba: Email y contraseña correctos deben autenticar al usuario

⚠️ Caso límite: Verificar que retorna el usuario correcto

📝 Importancia: Es el flujo básico de acceso a la aplicación

✅ TEST 2: test_login_with_case_insensitive_email

❓ ¿Por qué?: Los usuarios escriben emails con mayúsculas inconsistentes

Prueba: "TEST@EXAMPLE.COM" debe funcionar igual que "test@example.com"

⚠️ Caso límite: Probamos TEST@EXAMPLE.COM, test@example.com, TeSt@ExAmPlE.cOm

📝 Importancia: Mejora la experiencia del usuario, no falla por mayúsculas

✅ TEST 3: test_login_with_wrong_password

❓ ¿Por qué?: Prevenir acceso no autorizado con contraseña incorrecta

Prueba: Contraseña incorrecta debe ser rechazada con mensaje de error

⚠️ Caso límite: Contraseña casi correcta pero diferente

📝 Importancia: Seguridad básica de la aplicación

✅ TEST 4: test_login_with_nonexistent_email

❓ ¿Por qué?: Evitar acceso con emails que no están registrados

Prueba: Email válido pero no registrado debe rechazarse

⚠️ Caso límite: Email con formato válido pero inexistente

📝 Importancia: Solo usuarios registrados pueden acceder

🔑 CLASE: TestSignUpForm

✅ TEST 5: test_successful_signup_with_valid_data

❓ ¿Por qué?: Usuarios nuevos deben poder registrarse correctamente

Prueba: Datos válidos crean un usuario nuevo en la BD

⚠️ Caso límite: Email único, contraseñas coinciden exactamente

📝 Importancia: Es el punto de entrada para nuevos usuarios

✅ TEST 6: test_signup_with_duplicate_email

❓ ¿Por qué?: Prevenir registros con emails duplicados en el sistema

Prueba: Email ya registrado debe ser rechazado con error específico

⚠️ Caso límite: Email exactamente igual al ya existente

📝 Importancia: Cada usuario tiene identidad única por email

✅ TEST 7: test_signup_with_duplicate_email_case_insensitive

❓ ¿Por qué?: "test@example.com" y "TEST@EXAMPLE.COM" son la misma persona

Prueba: La búsqueda de duplicados ignora mayúsculas/minúsculas

⚠️ Caso límite: Intenta registrar con mayúsculas si existe en minúsculas

📝 Importancia: Evita múltiples cuentas de la misma persona

✅ TEST 8: test_signup_with_mismatched_passwords

❓ ¿Por qué?: Las dos contraseñas ingresadas (password1, password2) deben coincidir

Prueba: Contraseña de confirmación diferente es rechazada

⚠️ Caso límite: Diferencia de 1 solo carácter, ambas vacías

📝 Importancia: Evita errores al tipear la contraseña

🐮 4. TESTS DE ANIMALS (Animales) - 10 TESTS

Módulo: tests/test_animals.py

CLASE: TestAnimalModel

✅ TEST 1: test_animal_creation

❓ ¿Por qué?: Validar que los animales se crean con todos los datos correctos

Prueba: Animal creado tiene código, especie, raza, sexo y fecha de nacimiento

⚠️ Caso límite: Campos requeridos vs opcionales están correctamente configurados

📝 Importancia: Base para todo seguimiento de animales

✅ TEST 2: test_animal_string_representation_with_codigo

❓ ¿Por qué?: Los animales deben mostrarse como "CÓDIGO - ESPECIE" en listados

Prueba: str(animal) retorna el formato correcto "TEST-001 - Vaca"

⚠️ Caso límite: Códigos especiales, especies con espacios

📝 Importancia: Mejora legibilidad en admin y listados

✅ TEST 3: test_animal_string_representation_without_codigo

❓ ¿Por qué?: Si no hay código, mostrar solo la especie

Prueba: str(animal) sin código retorna solo "Caballo"

⚠️ Caso límite: Código None (falta identificador)

📝 Importancia: Manejo de datos incompletos de forma elegante

✅ TEST 4: test_animal_sexo_choices

❓ ¿Por qué?: Solo M (Macho) o F (Hembra) son permitidos

Prueba: Validar que solo opciones válidas se aceptan

⚠️ Caso límite: M válido ✓, F válido ✓, X inválido ✗

📝 Importancia: Consistencia en filtros y reportes de sexo

✅ TEST 5: test_animal_unique_codigo

❓ ¿Por qué?: Cada animal debe tener un código único (no dos "VAC-001")

Prueba: Intentar crear segundo animal con mismo código genera error

⚠️ Caso límite: Código None permite múltiples (si no se especifica)

📝 Importancia: Identificación única de cada animal

📊 CLASE: TestAnimalAge

✅ TEST 6: test_animal_age_in_years

❓ ¿Por qué?: Calcular la edad en años para animales adultos

Prueba: Animal de 3 años se calcula correctamente como 3 años

⚠️ Caso límite: Exactamente 1 año, 1 año + 1 día

📝 Importancia: Información crítica para manejo del rebaño

✅ TEST 7: test_animal_age_in_months

❓ ¿Por qué?: Mostrar edad en meses para animales menores de 1 año

Prueba: Animal de 5 meses (150 días) se calcula como ~5 meses

⚠️ Caso límite: 1 mes exacto, 30 días, 60 días

📝 Importancia: Mejor precisión en animales jóvenes

✅ TEST 8: test_animal_age_in_days

❓ ¿Por qué?: Mostrar edad en días para animales muy jóvenes

Prueba: Animal de 15 días se calcula como 15 días

⚠️ Caso límite: 1 día, 15 días, 29 días (antes de 1 mes)

📝 Importancia: Precisión máxima para crías newborn

✅ TEST 9: test_animal_newborn

❓ ¿Por qué?: Animal nacido hoy debe mostrar 0 días exacto

Prueba: Fecha de nacimiento = hoy → edad = 0 días

⚠️ Caso límite: Nacimiento hoy vs ayer

📝 Importancia: Registrar crías recién nacidas correctamente

✅ TEST 10: test_animals_ordered_by_birthdate_descending

❓ ¿Por qué?: Mostrar animales más jóvenes (recientes) primero en listados

Prueba: Ordenamiento por -fecha_de_nacimiento (descendente)

⚠️ Caso límite: Múltiples animales, misma fecha, orden consistente

📝 Importancia: UX: Los animales nuevos aparecen al inicio

5. RESUMEN DE TESTS

Accounts: 8 tests ✅

- 4 tests de login

- 4 tests de signup

Animals: 10 tests ✅

- 5 tests del modelo

- 5 tests de cálculo de edad

TOTAL: 18 tests ✅

⚠️ 6. CASOS LÍMITE CUBIERTOS

✓ Validaciones de datos requeridos

✓ Case-insensitivity en emails (TEST@EXAMPLE.COM = test@example.com)

✓ Unicidad de identificadores (emails, códigos de animales)

✓ Opciones limitadas (sexo: solo M o F)

✓ Límites de caracteres (código máx 50 chars)

✓ Eliminación en cascada (lote eliminado → animales eliminados)

✓ Cálculos complejos (edad en años, meses, días)

✓ Autenticación y autorización

✓ Datos incompletos (código opcional en algunos casos)

✓ Ordenamiento automático (animales por fecha descendente)

📊 7. INTERPRETACIÓN DE RESULTADOS

✅ EJECUCIÓN EXITOSA:

$ pytest

===== 18 passed in 1.45s =====

✓ Todos los tests pasaron correctamente

❌ FALLOS:

$ pytest

===== 16 passed, 2 failed in 1.50s =====

✗ 2 tests fallaron → Ver detalles

🔍 VER DETALLES:

```
pytest -v           # Mostrar cada test
pytest -s           # Mostrar prints y excepciones
pytest --tb=long    # Traceback detallado
pytest -x           # Parar en primer fallo
```

8. CONCLUSIÓN

Con 18 tests enfocados en accounts y animals, el proyecto BIOS tiene:

✓ 🔐 Autenticación segura (login/signup con validaciones)

✓ 🆔 Identidades únicas (emails únicos, códigos únicos)

✓ 🐮 Gestión correcta de animales (datos completos y organizados)

✓ 🗓️ Cálculo preciso de edades (años, meses, días)

✓ 📝 Datos limpios y normalizados (mayúsculas, espacios)

✓ 🔍 Búsquedas y filtros funcionan correctamente

✓ 🗑️ Eliminación en cascada de datos relacionados

✓ 🚫 Protección de acceso por autenticación

Estos tests garantizan que los módulos críticos del sistema funcionan de forma confiable y segura.