

## PORTAL DE PROCESOS VEGA: INFORME DE PRUEBAS UNITARIAS (AVANCE IL1)

**Programa:** Programa Especializado en Ingeniería de Software

**Módulo:** Pruebas de Software y Calidad

**Docente:** Marco Manrique

**Participante:** Bustamante Guerra Juan Antonio

**Fecha:** 29 de Octubre de 2025

### Tabla de Contenidos:

1. Introducción
2. Diseño de Casos de Prueba Unitarios (Backend)
3. Configuración del entorno de pruebas
4. Desarrollo y Ejecución de Pruebas Unitarias
5. Validación de Cobertura de Código
6. Análisis de Resultados y Conclusiones
7. Anexos

#### 1. . Introducción

El presente informe detalla el diseño, configuración, desarrollo y ejecución de las **pruebas unitarias** para los componentes del **backend** del proyecto "Portal de Procesos Vega". Este portal, desarrollado con Django y Django Rest Framework, tiene como objetivo centralizar la documentación y procedimientos operativos de la empresa Vega.

Siguiendo los lineamientos del Logro 1 (IL1) del módulo de Pruebas de Software y Calidad, este avance se enfoca en validar el correcto funcionamiento aislado de los componentes clave implementados hasta la fecha, aplicando buenas prácticas y utilizando herramientas estándar de la industria para garantizar la calidad inicial del software. El objetivo principal es asegurar que las unidades de código individuales (modelos, serializadores, vistas) se comportan según lo esperado antes de proceder a pruebas de mayor nivel como las de integración.

#### 2. Diseño de Casos de Prueba Unitarios (Backend)

Se diseñaron 6 casos de prueba unitarios enfocados en validar aspectos críticos de las aplicaciones users (manejo de usuarios y roles) y processes (manejo de la información de procesos). Estos casos cubren la lógica de los modelos, la correcta serialización de datos (especialmente en la autenticación) y el comportamiento de las vistas bajo diferentes condiciones.

Tabla 1: Diseño de Casos de Prueba Unitarios (BACKEND - Portal Vega)

ID	Módulo	Archivo a Probar	Unidad a Probar	Objetivo	Pasos	Resultado Esperado
U-B-001	users ▾	models.py ▾	Modelo Profile (Rol por defecto)	Verificar que el rol por defecto de un nuevo perfil sea 'CAJERO'.	1. Crear User de prueba. 2. Verificar creación automática de Profile. 3. Leer profile.role.	El valor de role debe ser 'CAJERO'.
U-B-002	users ▾	models.py ▾	Modelo Profile ( <b>str</b> )	Asegurar que <code>str(profile)</code> sea informativo.	1. Crear User ('testuser', rol 'JEFE'). 2. Llamar a <code>str()</code> del Profile.	Debe devolver "testuser - Jefe".
U-B-003	processes ▾	models.py ▾	Modelo Proceso ( <b>str</b> )	Verificar que <code>str(proceso)</code> incluya tipo y título.	1. Crear Proceso ('Licitaciones', 'B2B'). 2. Llamar a <code>str()</code> del Proceso.	Debe devolver "[B2B] Licitaciones".
U-B-004	users ▾	serializ... ▾	MyTokenObtainPairSerializer (Payload Token)	Confirmar que el token JWT incluya username y role.	1. Crear User ('testjefe', rol 'JEFE'). 2. Usar serializer para generar token. 3. Decodificar payload.	El payload debe contener username: 'testjefe' y role: 'JEFE'.
U-B-005	processes ▾	views.py ▾	ProcesoViewSet (get_queryset - sin filtro)	Verificar que devuelva todos los procesos sin filtro.	1. Crear 2 B2B, 1 B2C. 2. Simular GET <code>/api/processes/</code> . 3. Ejecutar <code>get_queryset()</code> .	El queryset debe contener los 3 procesos.
U-B-006	processes ▾	views.py ▾	ProcesoViewSet (get_queryset - con filtro)	Verificar que el filtro <code>?tipo_venta=B2B</code> funcione.		

### 3. Configuración del entorno de pruebas

Para la ejecución de las pruebas unitarias del backend, se configuró un entorno basado en herramientas estándar y recomendadas para proyectos Django.

Herramientas Seleccionadas:

PyTest ( pytest-django): Se eligió PyTest por su sintaxis concisa, su potente sistema de accesorios y su amplia adopción en la comunidad Python. El complemento pytest-djangofacilita la integración con el framework Django, gestionando automáticamente la base de datos de prueba y el acceso a la configuración del proyecto.

Coverage.py: Herramienta estándar para medir la cobertura de código en Python. Se integra con PyTest para generar informes detallados sobre qué líneas de código son ejecutadas por las pruebas.

Instalación: Las herramientas se instalaron en el entorno virtual del proyecto ( venv) utilizando pip:

```
pip install pytest pytest-django coverage
```

Configuración de PyTest: Se creó el archivo backend/pytest.ini para indicar a PyTest cómo encontrar la configuración de Django y los archivos de prueba:

```
[pytest]
DJANGO_SETTINGS_MODULE = core.settings
python_files = tests.py test_*.py *_tests.py
```

Estructura de Archivos: Siguiendo las convenciones de Django, las pruebas unitarias se implementaron en archivos tests.py dentro de cada aplicación ( users/tests.py, processes/[tests.py](#)).

### 4. Desarrollo y Ejecución de Pruebas Unitarias

Se implementó el código para los 6 casos de prueba unitarios diseñados, utilizando las funcionalidades de pytest-django para interactuar con la base de datos de prueba y solicitudes similares

```
(venv) C:\Proyectos\Portal_Vega\backend>pytest users/tests.py::test_profile_default_role
===== test session starts =====
platform win32 -- Python 3.10.11, pytest-8.4.2, pluggy-1.6.0
django: version: 5.2.7, settings: core.settings (from ini)
rootdir: C:\Proyectos\Portal_Vega\backend
configfile: pytest.ini
plugins: django-4.11.1
collected 1 item

users\tests.py . [100%]

===== 1 passed in 1.86s =====
```

La ejecución completa del conjunto de pruebas se realizó mediante el comando pytest desde la raíz de la carpeta backend.

```

(venv) C:\Proyectos\Portal_Vega\backend>pytest
===== test session starts =====
platform win32 -- Python 3.10.11, pytest-8.4.2, pluggy-1.6.0
django: version: 5.2.7, settings: core.settings (from ini)
rootdir: C:\Proyectos\Portal_Vega\backend
configfile: pytest.ini
plugins: django-4.11.1
collected 6 items

processes\tests.py ... [ 50%]
users\tests.py ... [100%]

===== 6 passed in 4.04s =====

```

Resultado de Ejecución: Todas las 6 pruebas unitarias implementadas pasaron exitosamente , indicando que los componentes individuales evaluados funcionan según lo especificado en el diseño de casos.

## 5. Validación de Cobertura de Código

Se utiliza coverage.py para medir el porcentaje del código fuente del backend que fue ejecutado durante la corrida de las pruebas unitarias. Los comandos ejecutados fueron:

```

coverage run -m pytest
coverage report -m

```

```

(venv) C:\Proyectos\Portal_Vega\backend>coverage run -m pytest
===== test session starts =====
platform win32 -- Python 3.10.11, pytest-8.4.2, pluggy-1.6.0
django: version: 5.2.7, settings: core.settings (from ini)
rootdir: C:\Proyectos\Portal_Vega\backend
configfile: pytest.ini
plugins: django-4.11.1
collected 6 items

processes\tests.py ... [ 50%]
users\tests.py ... [100%]

===== 6 passed in 5.03s =====

```

## Resultados de Cobertura:

```
(venv) C:\Proyectos\Portal_Vega\backend>coverage report -m
```

Name	Stmts	Miss	Cover	Missing
core\__init__.py	0	0	100%	
core\settings.py	22	0	100%	
processes\__init__.py	0	0	100%	
processes\admin.py	3	0	100%	
processes\apps.py	4	0	100%	
processes\migrations\0001_initial.py	5	0	100%	
processes\migrations\__init__.py	0	0	100%	
processes\models.py	8	0	100%	
processes\serializers.py	6	0	100%	
processes\tests.py	42	0	100%	
processes\views.py	12	0	100%	
users\__init__.py	0	0	100%	
users\admin.py	1	0	100%	
users\apps.py	6	0	100%	
users\migrations\0001_initial.py	7	0	100%	
users\migrations\__init__.py	0	0	100%	
users\models.py	8	0	100%	
users\serializers.py	24	2	92%	41-42
users\signals.py	9	0	100%	
users\tests.py	30	0	100%	
TOTAL	187	2	99%	

## 6. Análisis de Resultados y Conclusiones

La implementación y ejecución de las 6 pruebas unitarias diseñadas para el backend del "Portal Vega" ha sido **exitosa** . Todas las pruebas pasaron, validando el comportamiento esperado de los modelos, serializadores clave y la lógica de filtrado de las vistas implementadas hasta la fecha.

El análisis de cobertura arrojó un **excelente 99%** , demostrando que las pruebas ejercitan la gran mayoría del código funcional. Las únicas líneas no cubiertas corresponden a un manejo de error marginal en el serializador de usuarios, cuya ausencia no afecta la validación de la lógica principal requerida para este avance.

Se concluye que los objetivos del Logro 1 (IL1) han sido cumplidos satisfactoriamente . Se ha demostrado la capacidad de diseño de casos de pruebas unitarias basadas en requerimientos, configurar un entorno de pruebas con PyTest y Coverage.py, desarrollar el código de dichas pruebas siguiendo buenas prácticas y analizar los resultados obtenidos, incluyendo la métrica de cobertura.

## 7. Anexos

Enlace a la Rama de GitHub: El código fuente de las pruebas unitarias desarrolladas se encuentra en la siguiente rama del repositorio del proyecto:

<https://github.com/JuanBustamante107517/portal-procesos-vega/tree/feature/pruebas-qa-il1>