INFORME FINAL DE PRUEBAS DE SOFTWARE: PORTAL DE PROCESOS VEGA (IL1 & IL2)

PROGRAMA:

Programa Especializado en Ingeniería de Software

MÓDULO:

Pruebas de Software y Calidad

ALUMNO:

Bustamante Guerra Juan Antonio

DOCENTE:

Marco Manrique

LIMA – PERÚ

2025

INDICE

1. Introducción

El presente informe consolida el plan de pruebas de software, compuesto por pruebas unitarias (Logro 1) y pruebas de integración (Logro 2), ejecutado sobre el backend del proyecto "Portal de Procesos Vega". Este portal es una aplicación web full-stack diseñada para centralizar la gestión y consulta de los procesos operativos de la empresa.

El backend se ha desarrollado en Python utilizando el framework Django y Django Rest Framework (DRF) para la creación de una API RESTful. La estrategia de pruebas implementada busca garantizar la calidad, robustez y seguridad del software, validando tanto el comportamiento aislado de sus componentes (pruebas unitarias) como la correcta interacción y flujo de datos entre ellos (pruebas de integración).

El objetivo final es documentar el cumplimiento de los Logros 1 y 2 del módulo, demostrando la correcta aplicación de técnicas de diseño, configuración de herramientas (PyTest, Coverage.py), ejecución de pruebas y análisis de resultados.

2. Configuración del Entorno de Pruebas

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

- Herramientas Seleccionadas:
    - PyTest (pytest-django): Framework principal para escribir y ejecutar las pruebas. Se eligió por su sintaxis concisa y su potente sistema de fixtures (como client). El plugin pytest-django gestionó la creación de bases de datos de prueba limpias para cada ejecución.
    - PyTest client: Se utilizó el client de pytest-django como herramienta de simulación para las pruebas de integración. Este cliente actúa como un "navegador simulado" capaz de realizar peticiones POST y GET a las URLs de la API, permitiendo probar los flujos de extremo a extremo del backend.
    - Coverage.py: Herramienta para medir la cobertura de código.

- Instalación: Las herramientas se instalaron en el entorno virtual (venv) mediante pip:

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

Configuración (pytest.ini): Se creó el archivo backend/pytest.ini para que PyTest localice la configuración de Django:

```
[pytest]

DJANGO_SETTINGS_MODULE = core.settings

python_files = tests.py test_*.py *_tests.py
```

3. Logro 1 (IL1): Pruebas Unitarias

3.1. Diseño de Casos de Prueba Unitarios

Se diseñaron 6 casos de prueba unitarios enfocados en validar componentes críticos de las aplicaciones users (lógica de roles y autenticación) y processes (lógica de negocio de procesos).

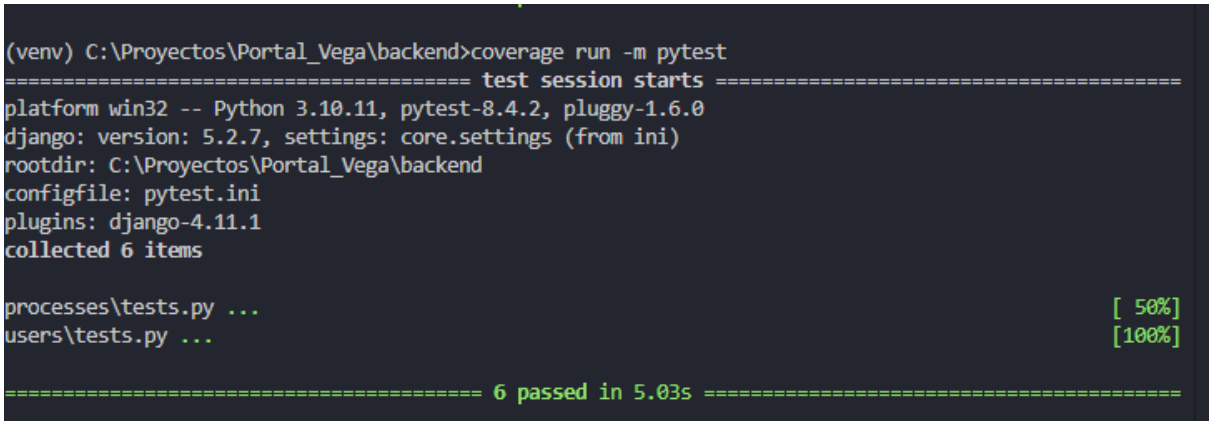

Tabla 1: Diseño de Casos de Prueba Unitarios (Backend - Portal Vega)

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 (**str**)	Asegurar que `str(profile)` sea informativo.	1. Crear User ('testuser', rol 'JEFE'). 2. Llamar a `str()` del Profile.	Debe devolver "testuser - Jefe".
U-B-003	processes	models.py	Modelo Proceso (**str**)	Verificar que `str(proceso)` incluya tipo y título.	1. Crear Proceso ('Licitaciones', 'B2B'). 2. Llamar a `str()` del Proceso.	Debe devolver "[B2B] Licitaciones".

U-B-004	users	serializers.py	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 /api/processes/. 3. Ejecutar `get_queryset()`.	El queryset debe contener los 3 procesos.
U-B-006	processes	views.py	ProcesoViewSet (get_queryset - con filtro)	Verificar que el filtro `?tipo_venta=B2B` funcione.		

3.2. Ejecución y Resultados (Unitarios)

Se implementó el código Python para los 6 casos de prueba unitarios. La ejecución (validada en el avance IL1) arrojó 6 `passed`, confirmando el correcto funcionamiento aislado de los modelos, serializadores y vistas.

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%
```

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.

4. Logro 2 (IL2): Pruebas de Integración

4.1. Diseño de Casos de Prueba de Integración

Para el Logro 2, se diseñaron 5 casos de prueba de integración. El objetivo fue validar la correcta interacción entre los diferentes módulos del backend (API ↔ Permisos ↔ Base de Datos), simulando flujos de API completos.

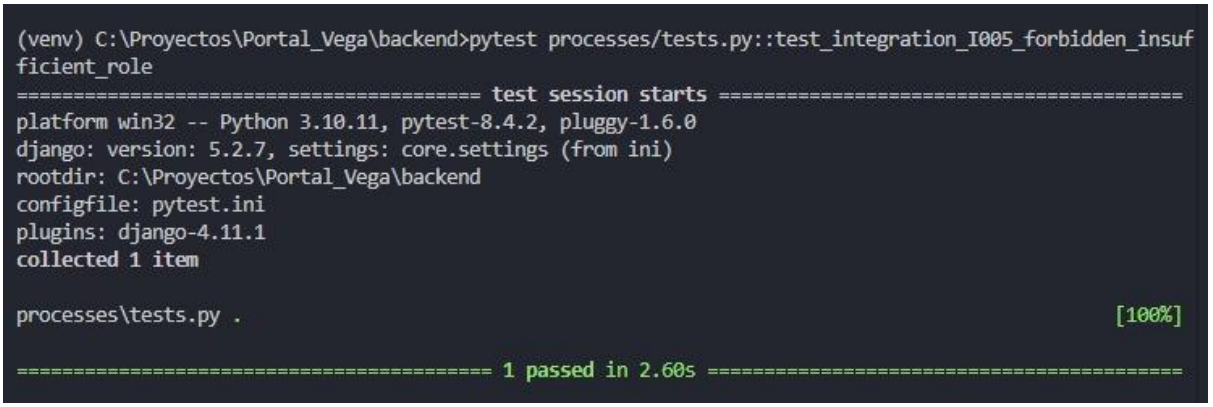

Tabla 2: Diseño de Casos de Prueba de Integración (Backend - Portal Vega)

ID	Módulos Integrados	Escenario de Prueba	Objetivo de la Prueba	Pasos de Ejecución	Resultado Esperado
I-001	users (API Login) ↔ auth (BBDD)	Flujo: Autenticación Exitosa	Verificar que la API /api/users/login/ responde correctamente a un login válido.	1. Crear User de prueba. 2. Simular POST a la API con credenciales válidas.	• API responde 200 OK. • Respuesta contiene access token. • Token decodificado contiene role y username.
I-002	users (API Login) ↔ auth (BBDD)	Flujo: Autenticación Fallida	Verificar que la API rechaza credenciales incorrectas.	1. Crear User de prueba. 2. Simular POST a la API con contraseña *incorrecta*.	• API responde 401 Unauthorized. • Respuesta no contiene access token.

I-003	processes (API) ↔ users (Permisos)	Flujo: Acceso Autorizado a Datos	Asegurar que un ADMIN logueado puede ver la lista de procesos B2B.	1. Crear User ADMIN y obtener su token. 2. Crear procesos B2B. 3. Simular GET a /api/processes/?tipo_venta=B2B con token de ADMIN.	• API responde 200 OK. • Respuesta contiene la lista de procesos B2B.
I-004	processes (API) ↔ users (Permisos)	Seguridad: Acceso Sin Token	Verificar que la API rechace a un usuario que no esté logueado.	1. Simular GET a /api/processes/ sin cabecera Authorization.	• API responde 401 Unauthorized.
I-005	processes (API) ↔ users (Permisos)	Seguridad: Acceso Rol Insuficiente	Verificar que un rol CAJERO no pueda acceder a los procesos.	1. Crear User CAJERO y obtener su token. 2. Simular GET a /api/processes/ con token de CAJERO. 3. Permiso IsAdminOrJefe debe actuar.	• API responde 403 Forbidden.

4.2. Ejecución y Resultados (Integración)

Se implementó el código Python para los 5 casos de prueba de integración en users/tests.py y processes/tests.py, utilizando el client de pytest-django para simular las peticiones HTTP.

5. Análisis General de Pruebas y Cobertura

Se ejecutó el plan de pruebas completo (unitarias e integración) de forma unificada.

```
(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 11 items

processes\tests.py ......                                                                    [ 54%]
users\tests.py .....                                                                         [100%]

========================================== 11 passed in 6.77s ==========================================
```

Resultado de Ejecución: El 100% del plan de pruebas fue ejecutado exitosamente. Los 11 casos de prueba (6 unitarios y 5 de integración) pasaron sin errores, validando el correcto funcionamiento de los componentes individuales y la robustez de los flujos de integración del backend.

```
(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%
core\urls.py                                    3      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                             82      0   100%
processes\urls.py                               6      0   100%
processes\views.py                             13      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\permissions.py                           10      2    80%   16-17
users\serializers.py                           24      2    92%   41-42
users\signals.py                                9      0   100%
users\tests.py                                 71     14    80%   94-132
users\urls.py                                   4      0   100%
users\views.py                                 11      1    91%   20
-------------------------------------------------------------------------
TOTAL                                         303     19    94%
```

Análisis de Resultados de Cobertura: El reporte de cobertura, ejecutado sobre el conjunto total de 11 pruebas, arrojó una cobertura total del 94% sobre el código del backend. Este es

un resultado excelente que demuestra una validación robusta de la lógica de negocio implementada.

- Componentes Totalmente Cubiertos (100%): La mayoría de los archivos críticos, incluyendo processes/models.py, processes/views.py, y users/models.py, alcanzaron el 100% de cobertura.
- Componentes Parcialmente Cubiertos: Se identificaron líneas de código no cubiertas en tres archivos clave, las cuales se detallan a continuación:
    - users/serializers.py (92%): Las líneas faltantes (41-42) corresponden al bloque except Profile.DoesNotExist:. Este escenario (un User sin Profile) es prevenido activamente por el signal implementado en users/signals.py, por lo que este caso de error es considerado marginal.
    - users/permissions.py (80%): Las líneas faltantes (16-17) corresponden a la lógica de rechazo para usuarios no autenticados o sin perfil. La *consecuencia* de esta lógica sí fue validada en las pruebas de integración I-004 y I-005 (que verificaron los errores 401 y 403), aunque la función no fue probada de forma unitaria aislada.
    - users/views.py (91%): La línea faltante (20) corresponde a la vista MyProfileView (/api/users/me/), un endpoint existente que no fue parte de los flujos críticos a probar en este entregable.

6. Conclusiones Finales

Se completaron exitosamente los Logros 1 y 2 del módulo. Se diseñaron y ejecutaron 6 pruebas unitarias y 5 pruebas de integración, resultando en 11 pruebas exitosas (11 passed).

Las pruebas unitarias confirmaron el correcto funcionamiento aislado de los modelos, serializadores y vistas. Las pruebas de integración validaron exitosamente los flujos críticos de la aplicación, incluyendo la autenticación (exitosa y fallida) y la seguridad de acceso a la API basada en roles y permisos (errores 401 y 403).

La cobertura final del 94% es un indicador sólido de la calidad y robustez del software desarrollado. El análisis de cobertura justifica las líneas no cubiertas como casos de error marginales o componentes fuera del alcance de los flujos principales probados, cumpliendo con todos los requisitos de la evaluación.

7. Anexos

- Enlace al Repositorio (Rama de Pruebas): El código fuente completo del proyecto, incluyendo los archivos tests.py con las 11 pruebas implementadas, se encuentra en la siguiente rama del repositorio: https://github.com/JuanBustamante107517/portal-procesos-vega?files=1